Available online at www.sciencedirect.com
SCIENCE DIRECT®
Artificial Intelligence 170 (2006) 507–541
Artificial Intelligence
www.elsevier.com/locate/artint

Conformant planning via heuristic forward search: A new approach ☆

Jörg Hoffmann [a,*], Ronen I. Brafman [b]

[a] *Max Planck Institute for Computer Science, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany*
[b] *Department of Computer Science, Ben-Gurion University, PO Box 653, Beer-Sheva 84105, Israel*

Received 16 July 2004; received in revised form 2 December 2005; accepted 4 January 2006

Available online 9 February 2006

Abstract

Conformant planning is the task of generating plans given uncertainty about the initial state and action effects, and without any sensing capabilities during plan execution. The plan should be successful regardless of which particular initial world we start from. It is well known that conformant planning can be transformed into a search problem in belief space, the space whose elements are sets of possible worlds. We introduce a new representation of that search space, replacing the need to store sets of possible worlds with a need to reason about the effects of action sequences. The reasoning is done by implication tests on propositional formulas in conjunctive normal form (CNF) that capture the action sequence semantics. Based on this approach, we extend the classical heuristic forward-search planning system FF to the conformant setting. The key to this extension is an appropriate extension of the relaxation that underlies FF's heuristic function, and of FF's machinery for solving relaxed planning problems: the extended machinery includes a stronger form of the CNF implication tests that we use to reason about the effects of action sequences. Our experimental evaluation shows the resulting planning system to be superior to the state-of-the-art conformant planners MBP, KACMBP, and GPT in a variety of benchmark domains.

Keywords: Planning under uncertainty; Heuristic search planning; Relaxed plan heuristic

1. Introduction

Conformant planning is the task of generating plans given uncertainty about the initial state and action effects, and without any sensing capabilities during plan execution. The plan should be successful (achieve all goal propositions) regardless of which particular initial world we start from and which action effects occur.

Conformant planning can be transformed into a search problem in the space of belief states, i.e., the space whose elements are sets of possible world states. This way, our uncertainty about the true current world state is modeled via the set of world states that we consider possible at this time. (Throughout the paper, we stick to this distinction between *world* states and *belief* states.) Bonet and Geffner [1] introduced the idea of planning in belief space using heuristic

☆ A preliminary version of this paper appears in [R. Brafman, J. Hoffmann, Conformant planning via heuristic forward search: A new approach, in: S. Koenig, S. Zilberstein, J. Koehler (Eds.), Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04), Whistler, Canada, Morgan Kaufmann, San Mateo, CA, 2004, pp. 355–364].

* Corresponding author.
E-mail addresses: hoffmann@mpi-sb.mpg.de (J. Hoffmann), brafman@cs.bgu.ac.il (R.I. Brafman).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2006.01.003

VISIT…

forward search. But the number of possible worlds in a belief state is typically large—even in simple examples like the infamous Bomb-in-the-toilet problem, there are exponentially many worlds in the initial belief state—and so Bonet and Geffner's system GPT, which explicitly represents these states, usually fails to scale up. In a number of planners, Bertoli, Cimatti, and Roveri [2,3] tackle this problem by using BDDs to represent the belief states. This often works better, but the size of the constructed BDDs is still prohibitive for scaling in many cases.[1]

In this paper we suggest a third, lazy, approach to represent belief states. The approach assumes, in its current form, that the goal and action preconditions are restricted to simple *conjunctions* of propositions, as in STRIPS.[2] We observe that, for conformant planning in this setting, it suffices to know the propositions that are true in the *intersection* of the worlds contained in a belief state—only these propositions will be true no matter what initial world we start from. An action sequence is a conformant plan iff it leads to a belief state where the intersection of the worlds contains all goal propositions. In our framework, a belief state s is represented just by the initial belief state representation together with the action sequence *act* that leads to s. To check for fulfillment of the goal and of action preconditions, we then test, for each proposition p, if it is contained in the intersection of the worlds in s, i.e., if p is always true after executing *act*. We say that such propositions are *known* in s. Doing the test for a proposition is hard in general; the corresponding decision problem is easily shown to be co-NP complete. Our implementation does the test by checking whether the proposition is implied by a CNF formula that captures the semantics of the action sequence.

In comparison to the belief state representations used in GPT and MBP, our approach trades space for time: assume the planner faces a belief state s and an applicable action a, and wants to know what the outcome s' of applying a to s would be. If, like in GPT and MBP, the planner maintains complete knowledge of s in memory then the computation can be done based exclusively on that knowledge. In our approach, however, the planner has no knowledge about s in memory other than the known propositions (and the path to s). To determine the known propositions for s', the planner has to reason about the entire action sequence leading to s', *including those actions that lead to* s. Our intuition is that modern SAT solvers, which can easily handle problem instances with tens of thousands of variables and clauses, will have little trouble reasoning about the effects of an action sequence if the possible interactions are not overly complex. Indeed, this intuition is supported quite impressively by excellent results we obtained in a variety of benchmark domains, using a largely naive DPLL [4,5] implementation as the underlying SAT solver.

While our representation of belief states is relatively efficient, a blind traversal of the search space is still infeasible due to the combinatorial explosion in possible action sequences. In classical planning (with no uncertainty about the initial state or action effects), this problem has been tackled most successfully by using heuristic functions based on a relaxation of the planning task, where the relaxation is to assume that all delete lists are empty. To each search state during, e.g., a forward search, a relaxed plan is computed. The length of the relaxed plan provides an, often very informative, estimate of the state's goal distance. FF [6] is one particularly successful system that uses this technique. Beside the search space representation, the main contribution of our work is the adaptation of the relaxation, and of FF's heuristic function, to the conformant setting. We discuss different options how to extend the relaxation to the conformant setting. After choosing one of these options, we extend FF's relaxed planning process to handle such relaxed conformant problems. We enrich FF's machinery with a stronger form of the CNF reasoning that is done to determine the known propositions in the belief states. In a nutshell, the stronger reasoning corresponds to a 2-CNF projection of the real formula that would be needed to decide about known propositions during relaxed planning. Relative to that formula, the reasoning is thus complete, but not sound. We prove that the reasoning is complete *and* sound relative to our extended relaxation.

Our implementation, which we call Conformant-FF, is based on the FF code, and uses the same overall search arrangement as FF. We experimentally compare Conformant-FF with the state-of-the-art conformant planners MBP and KACMBP [3]—the latter is a conformant planner that has more advanced conformant planning heuristics than MBP. In a number of traditional conformant benchmarks, our experiments show Conformant-FF to be competitive: it often outperforms MBP, but is generally not as efficient as KACMBP. On the positive side, our approach demonstrates the potential to *combine* the strengths of FF with conformant abilities in domains that combine classical and conformant aspects. In a number of classical planning benchmarks enriched with uncertainty, Conformant-FF shows fine scalability, dramatically outperforming MBP as well as KACMBP. We also run limited experiments with GPT

[1] It should be noted that GPT and the BDD-based MBP planner are capable of doing much more than conformant planning.

[2] Extension to handle disjunctions is conceptually easy; we outline in the conclusion of the paper how to do so.

and with POND [7], a recent conformant planner using techniques related to Conformant-FF. The results indicate that, in many domains, Conformant-FF outperforms these planners too.

In our current implementation of Conformant-FF, we only deal with uncertainty about the initial state, i.e. we do *not* deal with non-deterministic effects. The extensions necessary to make Conformant-FF handle such effects are conceptually easy, with one important exception, namely repeated states checking. While there is a straightforward extension for repeated states checking in the presence of non-deterministic effects, it is more than doubtable that this extension would work well, i.e., have reasonable pruning power, in practice. We have not yet worked out an extended technique that appears practical. We leave this open for future work. To inform the reader about the main issues in this important aspect of Conformant-FF's further development, we include an extra section on non-deterministic effects. In the other parts of the paper, only uncertainty about the initial state is considered.

The paper is organized as follows. Section 2 briefly describes the planning framework we consider. Section 3 describes the search space representation and the computation of known propositions. Section 4 discusses conformant relaxations and explains our extension of FF's heuristic function (this is the technically most challenging part of our work, and the section forms more than half of the technical part of this paper). Section 5 explains how we avoid repeated search states. Section 6 gives some implementation details and our empirical results. Section 7 discusses non-deterministic effects, Section 8 discusses related work. Section 9 concludes the paper and gives an outlook on future work. All non-trivial proofs are moved into Appendix A, and are replaced in the text by proof sketches, to improve readability.

2. Planning background

The conformant planning framework we consider adds uncertainty to a subset of the classical ADL language. The subset of ADL we consider is (sequential) STRIPS with conditional effects, i.e., the following. Propositional planning tasks are triples (A, I, G) corresponding to the *action set*, *initial world state*, and *goals*. I and G are sets of propositions. Actions a are pairs $(pre(a), E(a))$ of the *precondition*—a set of propositions—and the *effects*—a set of conditional effects. A conditional effect e is a triple $(con(e), add(e), del(e))$ of proposition sets, corresponding to the effect's *condition*, *add*, and *delete* lists respectively (for conformant planning, one needs conditional effects as otherwise the same action sequence can hardly achieve the goal from different initial worlds). An action a is *applicable* in a world state w if $pre(a) \subseteq w$, i.e., if the world state satisfies all of a's preconditions. If a is not applicable in w, then the result of applying a to w is undefined. If a is applicable in w, then all conditional effects $e \in E(a)$ get executed whose condition satisfies $con(e) \subseteq w$ (unconditional effects have $con(e) = \emptyset$); we say that such effects *occur*. Executing a conditional effect e in w results in the world state $w - del(e) + add(e)$. We require that actions are not self-contradictory, i.e., if there is a proposition p such that, in world state w, $p \in add(e)$ and $p \in del(e')$ for two effects e and e' that both occur, then the result of applying the respective action is undefined. An action sequence is a *plan* (a solution) if the world state that results from iterative execution of the actions, starting in the initial world state, leads to a *goal state*, i.e. to a world state that contains all the goals.

The conformant planning setting we consider extends the above with uncertainty about the initial state. The initial state is now a belief state that is represented by a propositional CNF formula $\mathcal{I}$. The possible initial world states are those that satisfy that formula. Slightly abusing the powerset notation, we denote the (possibly exponentially large) set of the possible initial world states with $2^{\mathcal{I}}$. An action sequence is called *executable* if all actions in it are applicable at their point of execution no matter what initial world state one starts from. An action sequence $act \in A^*$ is a *plan* for a task $(A, \mathcal{I}, G)$ if, for any possible initial world state $I \in 2^{\mathcal{I}}$, executing act in I results in a goal state. (Note that plans are executable by this definition.)

By saying that the result of applying non-applicable actions is undefined, we require that all actions in the plan must be applicable at their point of execution no matter what the initial world state is. An alternative definition is to say that, if an action is not applicable in a world state, then applying the action does not change that world state. This alternative definition is of theoretical interest for our heuristic function (Section 4.1), and we call it *generous*. If an action sequence act solves a task $(A, \mathcal{I}, G)$ according to the generous definition of the result function, then act is called a *generous plan* for $(A, \mathcal{I}, G)$. Note that, given a task $(A, \mathcal{I}, G)$, if for all actions a and effects $e \in E(a)$ one includes $pre(a)$ into $con(e)$, and then sets $pre(a) := \emptyset$, then the resulting task under the non-generous semantics

corresponds exactly to the original task under the generous semantics. So the generous semantics can be simulated with the non-generous semantics and using the latter does not make us lose generality.[3]

3. Search space

As explained in the introduction, we perform a forward search in belief space. The search states are belief states. A belief state s is represented by the initial state representation $\mathcal{I}$ together with the action sequence *act* that leads from the initial state to s. In line with our definition of action semantics, during search we make sure that every action in the sequence will be applicable, no matter what the initial world state is; that is, we ensure that every considered action sequence is executable. The reasoning necessary for this, as well as for detecting goal belief states, is based on the computation of known propositions.

For each belief state encountered during search (including the initial belief state), we compute the sets of known and negatively known propositions. These are defined as follows. Given a conformant planning task $(A, \mathcal{I}, G)$, a belief state s corresponding to an executable action sequence $act \in A^*$, and a proposition p, we say that p is *known* in s if, for all $I \in 2^{\mathcal{I}}$, executing *act* in I results in a world state that contains p. We say that p is *negatively known* in s if for all $I \in 2^{\mathcal{I}}$ executing *act* in I results in a world state that does not contain p (knowing the propositions that will always be false helps speed up the reasoning, see below). A proposition that is neither known nor negatively known is *unknown*. Deciding about whether a proposition is known or not is co-NP complete.

Proposition 1. *Given a conformant planning task $(A, \mathcal{I}, G)$, a belief state s represented by an action sequence $act \in A^*$, and a proposition p. Deciding whether p is known in s is co-NP complete.*

Proof. This result essentially follows from Theorem 2 of [9]. The following is a direct argument: We consider the complementary problem. Membership in NP: non-deterministically guess an initial world state, and check if it satisfies $\mathcal{I}$, and if p does not hold upon execution of *act*. NP-hardness follows by a trivial reduction from SAT. Given a propositional CNF formula ϕ, just take that formula to be the initial formula $\mathcal{I}$, and ask if some proposition p not contained in ϕ is not known in the state that results from executing the empty action sequence. This is the case, by definition, iff ϕ is satisfiable: p can only be known if $2^{\mathcal{I}}$ is empty. □

Note that the proposition holds independently of the allowed goal conditions, and that the hardness result holds even if there are no actions at all. The hardness of the decision problem follows already from the fact that the initial state is a CNF formula.

We compute the sets of known and negatively known propositions in a search (belief) state by using a CNF corresponding to the semantics of the respective (executable) action sequence as follows. We use a time index to differentiate between values of propositions at different points along the execution of the action sequence. Say the action sequence is $act = \langle a_1, \ldots, a_n \rangle$. We obtain our CNF $\phi(act)$ as follows. We initialize $\phi(act)$ as $\mathcal{I}$ indexed with time 0 (i.e., for each clause $l_1 \vee \cdots \vee l_k$ in $\mathcal{I}$ we add $l_1(0) \vee \cdots \vee l_k(0)$ into $\phi(act)$). We then use a_1 to extend $\phi(act)$:

- *Effect Axioms*. For every effect e of a_1, $con(e) = \{c_1, \ldots, c_k\}$, and every proposition $p \in add(e)$, we insert the *add* effect axiom clause $\neg c_1(0) \vee \cdots \vee \neg c_k(0) \vee p(1)$. For every proposition $p \in del(e)$, we insert the *delete* effect axiom clause $\neg c_1(0) \vee \cdots \vee \neg c_k(0) \vee \neg p(1)$.
- *Frame Axioms*. For every proposition p, let $e_1, \ldots, e_n$ be the effects of a_1 such that $p \in del(e_i)$. For every tuple $c_1, \ldots, c_n$ such that $c_i \in con(e_i)$ we insert the *positive* frame axiom clause $\neg p(0) \vee c_1(0) \vee \cdots \vee c_n(0) \vee p(1)$. (Read this clause as an implication: if p was true before and has not been deleted by either of e_i, it is still true after a_1.) Symmetrically, when $e_1, \ldots, e_n$ are the effects of a_1 such that $p \in add(e_i)$, we insert for every tuple $c_1, \ldots, c_n$ with $c_i \in con(e_i)$ the *negative* frame axiom clause $p(0) \vee c_1(0) \vee \cdots \vee c_n(0) \vee \neg p(1)$ (if p was false before and has not been added, it is still false after a_1).

[3] One can also simulate the non-generous semantics with the generous semantics, by introducing a new goal proposition that is true initially, and that gets deleted whenever an action is executed whose precondition is not satisfied [8].

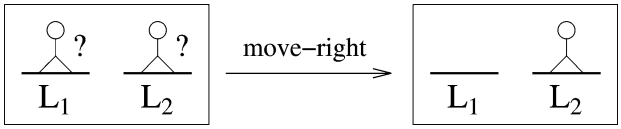

Fig. 1. A simple example task.

In the same fashion, we use a_2 to further extend the formula and so on until the axioms for a_n have been inserted. Note that, if one of the e_i that deletes/adds p has an empty effect condition, then no positive/negative frame axioms are inserted for p.[4] The resulting CNF $\phi(act)$ captures the semantics of *act* in the following sense.

Proposition 2. *Given a conformant planning task* $(A, \mathcal{I}, G)$*, and an executable n-step action sequence* $act \in A^*$*. Say we have a possible initial world state* $I \in 2^{\mathcal{I}}$ *and a proposition* p*. Then there is exactly one satisfying assignment* σ *to* $\phi_I(act)$ *(*$\phi(act)$ *where all variables at time* 0 *have been set to their values in* I*), and* p *holds upon execution of act in* I *iff* $\sigma(p(n)) = TRUE$*.*

Proof. Given fixed values for all variables at time 0, the axioms explicitly enforce a value for every proposition at every time step—either it is a (conditional) effect or its value remains the same due to an appropriate frame axiom. The enforced proposition values correspond exactly to the values that the respective propositions take on at the respective points in the plan. □

Note that $\phi(act)$ is basically a description of *act*'s semantics in the situation calculus [10]. The formula does not need to talk about action preconditions since *act* is assumed to be executable. (If *act* is not executable then Proposition 2 does not hold because a non-applicable action will result in an undefined state, but not affect the satisfying assignment to the formula.)

Proposition 2 immediately leads to the following simple result, which we use to compute the sets of known propositions.

Proposition 3. *Given a conformant planning task* $(A, \mathcal{I}, G)$*, a belief state* s *represented by an executable n-step action sequence* $act \in A^*$*, and a proposition* p*. Then* p *is known in* s *iff* $\phi(act)$ *implies* $p(n)$*.*

Proof. The formula $\phi(act) \wedge \neg p(n)$ is unsatisfiable, with Proposition 2, iff there is no $I \in 2^{\mathcal{I}}$ such that p does not hold upon executing *act* in I. □

We remark that Proposition 3 is not a new result; it can be obtained using Theorem 2 of [9]. The proof via Proposition 2 is kept in this paper just for reasons of self-containedness.

We use Proposition 3 to compute the set of known propositions as follows. Start with the empty set. Then, for each proposition p, hand $\phi(act) \wedge \neg p(n)$ over to the underlying SAT solver. If the result is "unsat" then add p to the known propositions. If the result is "sat", do nothing. Symmetrically, we compute the set of negatively known propositions by handing the formulas $\phi(act) \wedge p(n)$ over to the SAT solver.

Example 1. Say we have a robot that is initially at one out of two locations, modeled as $\mathcal{I} = \{at\text{-}L_1 \vee at\text{-}L_2, \neg at\text{-}L_1 \vee \neg at\text{-}L_2\}$. Our goal is to be at L_2, and we have a *move-right* action that has an empty precondition, and the conditional effect $(con = \{at\text{-}L_1\}, add = \{at\text{-}L_2\}, del = \{at\text{-}L_1\})$. A graphical sketch of this example is given in Fig. 1; the example will be re-used throughout the paper.

The known propositions in the search state s corresponding to the sequence $act = \langle move\text{-}right \rangle$ are computed as follows. The formula $\phi(act)$ consists of the following clauses:

[4] Note also that the number of frame axioms is exponential in the number of distinct conditional effects of a single action that can add/delete the same proposition. One can avoid this by introducing, for each time point t, a new proposition $p(e)(t)$ for each conditional effect e of the action applied at t, and ensuring that $p(e)(t)$ is true iff e occurs at time t. Single frame axioms of the form $\neg p(t) \vee p(e_1)(t) \vee \cdots \vee p(e_n)(t) \vee p(t+1)$ then suffice. We do not do this because in practice actions rarely affect the same proposition with different effects.

- $at\text{-}L_1(0) \vee at\text{-}L_2(0)$ and $\neg at\text{-}L_1(0) \vee \neg at\text{-}L_2(0)$: initial state formula.
- $\neg at\text{-}L_1(0) \vee at\text{-}L_2(1)$: add effect axiom for *move-right*.
- $\neg at\text{-}L_1(0) \vee \neg at\text{-}L_1(1)$: delete effect axiom for *move-right*.
- $\neg at\text{-}L_1(0) \vee at\text{-}L_1(0) \vee at\text{-}L_1(1)$: positive frame axiom for $at\text{-}L_1$. The axiom says that, if $at\text{-}L_1$ is true at 0, and the delete effect of *move-right* does not occur, then $at\text{-}L_1$ is still true at 1. Note that this cannot hold (since the delete effect occurs if $at\text{-}L_1$ is true), and the clause can be skipped.
- $\neg at\text{-}L_2(0) \vee at\text{-}L_2(1)$: positive frame axiom for $at\text{-}L_2$.
- $at\text{-}L_1(0) \vee \neg at\text{-}L_1(1)$: negative frame axiom for $at\text{-}L_1$.
- $at\text{-}L_2(0) \vee at\text{-}L_1(0) \vee \neg at\text{-}L_2(1)$: negative frame axiom for $at\text{-}L_2$.

To check whether $at\text{-}L_1$ is known in s, a satisfiability test is made on $\phi(act) \wedge \neg at\text{-}L_1(1)$. The result is "sat": a satisfying assignment σ is, e.g., that corresponding to $I = \{at\text{-}L_2\}$, i.e., $\sigma(at\text{-}L_2(0)) = TRUE$, $\sigma(at\text{-}L_1(0)) = FALSE$, $\sigma(at\text{-}L_2(1)) = TRUE$, $\sigma(at\text{-}L_1(1)) = FALSE$. Checking whether $at\text{-}L_2$ is known in s succeeds, however: $\phi(act) \wedge \neg at\text{-}L_2(1)$ is unsatisfiable. Inserting $\neg at\text{-}L_2(1)$ into the positive frame axiom for $at\text{-}L_2$ we get $\neg at\text{-}L_2(0)$, inserting $\neg at\text{-}L_2(1)$ into the effect axiom for *move-right* we get $\neg at\text{-}L_1(0)$, in consequence the initial state clause $at\text{-}L_1(0) \vee at\text{-}L_2(0)$ becomes empty. Similarly, one can find out that $at\text{-}L_1$ is negatively known in s.

The observation made in Proposition 3 gets us around enumerating all possible initial world states for computing whether a given proposition is known upon execution of *act* or not. While we *do* need to perform worst-case exponential reasoning about the formula $\phi(act)$, our empirical results show that this reasoning is feasible, as the interactions between action effects in practice (at least as reflected by our benchmark domains) are not overly complex.

Note that one can apply several significant reductions to the number of SAT calls made, and the size of the CNF formulas looked at. In our current implementation, these are:

- Simplify $\phi(act)$ by inserting the values of propositions at times $i < n$ which are known to be true or false—these values are stored (for the respective belief states) along the path corresponding to *act*. In effect, $\phi(act)$ only contains variables whose value is unknown at the respective points of *act*'s execution.
- Make SAT calls only on propositions p such that p may change its truth value due to a conditional effect e that *possibly* occurs: all of e's condition propositions are either known or unknown, and at least one of them is unknown. (For the initial belief state, SAT calls can be avoided for propositions occurring in unary clauses.)

Once the known propositions in s are computed, we can easily check if *act* is a plan: this is the case iff all goal propositions are known in s; similarly for action preconditions, see below. Note that one could check entailment of the goal (of an action precondition) with a single SAT call by negating the goal (the precondition), and conjoining the resulting disjunction with $\phi(plan)$. Decomposing this test into several tests for the individual propositions has the advantage that the knowledge about the propositions can be re-used for other tests.

The actions that are applicable to s, and that are used to generate the successor states, are those actions whose preconditions are all known in s, and whose effects cannot be self-contradictory in s. To make sure of the latter, we (potentially) do some additional SAT tests. Two effects e and e' are potentially self-contradictory if $add(e) \cap del(e') \neq \emptyset$ or $add(e') \cap del(e) \neq \emptyset$. For all such pairs e and e' of effects of each action whose preconditions are known in s, we check whether e and e' occur together in s for a possible initial world state. The check is made by a SAT call on the formula $\phi(act) \wedge \bigwedge_{c \in con(e) \cup con(e')} c(n)$. If the result is "sat" then e and e' can occur together and we skip the action. Of course, one can avoid unnecessary SAT calls when the known and negatively known propositions in s already imply that a pair e and e' of effects will or will not occur together.

4. Heuristic function

In classical planning, a successful idea has been to guide (forward, e.g.) search by heuristic functions based on a relaxation of the planning task, where the relaxation is to assume that all delete lists are empty. In every world state w during the search, the relaxed task starting from w is solved, and the length of the relaxed plan is taken to be the heuristic value to w. We now extend this idea to the conformant setting. This extension forms the technically most challenging part of our work, so we discuss it in detail.

This section is structured as follows. In Section 4.1, we discuss the advantages and drawbacks of possible options for an extended relaxation, choosing one of them for use in our system. In Section 4.2, we adapt the algorithm used by FF to solve relaxed tasks, and we prove that the adapted algorithm is sound and complete relative to the (chosen) extended relaxation. In Section 4.3, we explain two optimizations of the algorithm that we use in our implementation, in Section 4.4 we discuss two variations of the algorithm that we have also implemented.

4.1. Conformant relaxations

In the design of a heuristic function, two main decisions have to be made. The first of these is whether the heuristic should be *admissible*.[5] Admissible heuristics can be used to find optimal solutions using A*. In our work, as in the original FF system we sacrifice admissibility, and with that optimality, for runtime performance (see also below).

The second design decision to be taken in the creation of a heuristic function is a trade-off between the *informativity* and the computational efficiency of that function.[6] The more time we invest into reasoning inside the heuristic function, the more informative will that function be; at the extreme ends we have an exact heuristic function solving the entire problem to obtain its return value, and a trivial heuristic function returning always the same value.

In pure STRIPS planning, if we ignore the delete lists then the complexity of deciding plan existence goes down from PSPACE to P [11]. The complexity of deciding *bounded* plan existence is still NP-hard without delete lists, so it would be impractical to compute a provably shortest relaxed plan, and with that an admissible heuristic function. But we *can* use this relaxation as the basis for a heuristic function, by computing a relaxed plan in polynomial time, and taking its length as the heuristic value—this is done in FF. In STRIPS, not much more variation seems to be possible in this particular trade-off between informativity and computational efficiency: it is unclear how to ignore only some of the delete lists without falling back into the general, PSPACE-hard, planning case.

In the conformant setting we consider here, the design decisions possible in defining the relaxation, and thereby the heuristic function, are much more interesting and manifold. First of all, as a simple consequence of Proposition 1, deciding plan existence is still co-NP-hard without delete lists, and indeed without any actions. If one wants to obtain a heuristic function computable in worst-case polynomial time, then there is no way around relaxing (not only the actions but also) the initial state formula. On the other hand, in our approach to belief space search, as described in the previous section, several SAT calls may be made for every single generated search state. These SAT calls are worst-case exponential but seem to be feasible in practice, and a priori there is no reason why the same successful use of SAT solvers as part of the computation should not be possible in the heuristic function. It is important to note here that computing optimal relaxed plans, while also "only" NP-complete in the classical setting, seems practically infeasible; at least no technique for doing it reasonably efficiently has yet been developed (published). So we will stick, in the conformant setting, to not trying to compute optimal relaxed plans.

To avoid the computational complexity arising from destructive interactions between action effects, we also definitely want to stick to ignoring the delete lists in our conformant relaxation. The design decision to be taken is what other simplifications/relaxations we make to the conformant task description.

First, we observe that deciding conformant plan existence with empty delete lists can be done by a single CNF satisfiability test *if one uses the generous action execution semantics*. Recall from Section 2 that, under the generous execution semantics, non-applicable actions do not affect the world state, rather than resulting in an undefined state.

Theorem 1 *(A^+-complexity). Given a conformant task $(A, \mathcal{I}, G)$ where the delete lists of all effects are empty. Deciding whether there exists a generous plan for $(A, \mathcal{I}, G)$ is co-NP-complete.*

Proof sketch. Hardness follows directly from Proposition 1. Membership can be shown by constructing a CNF formula that encodes the semantics of executing all actions in parallel m times, where m is the number of distinct conditional effects in the task. The formula implies the goals (at time m) iff there exists a generous plan for $(A, \mathcal{I}, G)$. □

The CNF formula construction in the proof does not work if one assumes a non-generous execution semantics of actions, because in this case one cannot easily model a "parallel" execution of all actions. Intuitively, the generous

[5] A heuristic function is called admissible if it always underestimates the distance to the nearest solution state.

[6] Informativity is an informal concept meaning the quality of the guidance information that a heuristic provides.

execution semantics gives us the possibility to "select", for every initial world state, appropriate subsets from the parallel action steps. It is an open question what the complexity of deciding plan existence is in the absence of delete lists with a non-generous execution semantics. We remark that Theorem 8(iv) of [12] describes a slightly more general case (e.g., delete lists are allowed) for which the existence of a plan of bounded size is D^P-complete. This implies that our problem is in D^P (the class of all languages that can be defined as the intersection between an NP and a co-NP language).

Theorem 1 suggests to relax conformant planning tasks (only) by ignoring the delete lists, and then use a single SAT call per search state to compute a relaxed plan.[7] We did not try this out in an implementation because it does not appear very promising, or at least poses a number of serious difficulties. First of all, the size of the formula to be considered is a square function in the number of (ground) action effects and will become very large when there are many actions. It seems likely that SAT reasoning about such a large formula for every search state will take a lot of time. One might be able to ameliorate the difficulty by not doing a single SAT check asking for goal fulfillment after m parallel action steps, but instead looking at a sequence of CNF formulas with $1, 2, \ldots$ steps, and stopping when the goal first becomes implied (or when a termination criterion is met). Still even with this approach the CNF formulas reasoned about will be large. Another serious difficulty arises from the fact that we are looking for a heuristic estimate of the *number of actions* needed to achieve the goal. The experience from classical planning tells us that the number of parallel (relaxed) action steps is typically a very bad estimate for this number [13]. So, say we have found a CNF for k parallel action steps so that the goals are implied. That is, we have proved by an unsatisfiability result that this CNF implies truth of the goals after the kth step. So how do we actually extract a relaxed plan? What we need to find is a (minimal, ideally) subset of the actions (of the actions at the time steps) that still implies goal fulfillment. This corresponds to a (minimal, ideally), unsatisfiable subset of our CNF. There are techniques that extract unsatisfiable subsets of a CNF from Davis-Putnam style refutation proofs [14], by keeping track of the clauses involved in the resolution sequence corresponding to that proof. It seems questionable if such techniques can be made to work reliably and efficiently enough for use in the context of the heuristic computations we consider here.[8]

In our first version of Conformant-FF, we designed and implemented a heuristic function from the other end of the possible spectrum, with a worst-case polynomial computation. The relaxation that we made on top of ignoring the delete lists was to consider only a 2-CNF-projection of the relevant CNF formulas. That is, we considered a sequence of CNF formulas as above with $1, 2, \ldots$ parallel action applications, selected only two literals out of each clause, and stopped—with success—when the 2-projected CNF implied the goals, or—with failure—when a simple termination criterion told us that no success would be possible in future iterations.[9] In the case of success, from the reasoning done to prove the implication in the 2-CNF setting a relaxed plan could easily be extracted.

The worst-case polynomial computation was made possible because the whole CNF-semantics of the task, *including* the initial state formula, was 2-projected. However, it turned out that 2-projection of the initial state formula yielded very uninformative heuristic values in a lot of cases. In many examples, the uncertainty about the initial state lies in that the values of a set of multiple-valued variables—the positions/internal states of a set of objects—are unknown. Modeling such uncertainty with boolean variables, for a multiple-valued variable with k values one gets (a set of binary clauses and) a clause with k literals in the initial state formula. For values of k larger than 2, ignoring all but 2 of the literals in this clause quickly results in very bad approximations to the semantics of the multiple-valued variable.

In our current version of Conformant-FF—i.e. in the version reported about in this paper—we use a relaxation in between the above two alternatives. We do not do any relaxations to the initial state formula, thus paying the price of full SAT checks against that formula in order to have the benefit of taking full account of the information it contains. We do, however, relax the actions in that we ignore the delete lists of their effects, and we also re-use

[7] If the SAT call comes to the conclusion that there is no relaxed plan then it follows that there is no real plan from the search state either, and the state can be skipped.

[8] At the very least, significant implementation challenges would have to be overcome. Also, there would probably be ambiguities in the extraction of the unsatisfiable subsets of the CNFs, i.e., decisions that can be taken arbitrarily and that significantly affect the size of the found subset. Such ambiguities would directly translate into ambiguities—arbitrary significant variations—in the returned heuristic values, which are likely to affect search efficiency.

[9] The termination criterion was the same as we currently use, see below. The implementation of the process made extensive use of information about propositions already known to be true at certain time steps, similarly to our current implementation, see below.

the 2-projection that we previously used for the clauses related to action effects. At a time step t, for an action a that is (known to be) applicable at t, for an effect $e \in E(a)$ with condition $con(e) = \{c_1, \ldots, c_k\}$, for $p \in add(e)$, the valid implication is $c_1(t) \wedge \cdots \wedge c_k(t) \rightarrow p(t+1)$. Our 2-projection selects only one of the conditions c_i for inclusion in the implication clause, i.e. we get an implication of the form $c_i(t) \rightarrow p(t+1)$ (c_i is selected arbitrarily as the first condition proposition in our internal numbering). Formally, this simplification comes down to a relaxation that ignores, for each action effect, the effect's delete list as well as all but one of the propositions in the effect's condition. Given a set A of actions, we denote by $|_1^+$ any function from A into the set of all actions, such that $|_1^+$ maps each action a to the same action but with empty delete lists and with all but one condition proposition of each effect removed. We call $|_1^+$ a *relaxation function* for A. By $A|_1^+$ we denote the action set resulting from applying $|_1^+$ to each action in A. For an action sequence act, by $act|_1^+$ we denote the sequence of the images of the actions under $|_1^+$. For a conformant planning task $(A, \mathcal{I}, G)$, we call $(A|_1^+, \mathcal{I}, G)$ a *relaxation* of $(A, \mathcal{I}, G)$. If $act|_1^+$ is a plan for $(A|_1^+, \mathcal{I}, G)$, then act is called a *relaxed plan* for $(A, \mathcal{I}, G)$.

The algorithms that we use to solve relaxed tasks are the topic of the next subsection. Put briefly, the advantage of ignoring all but one effect condition (on top of ignoring all delete effects) is that, with this relaxation, the dependencies that the actions induce between propositions (at time steps) come down to binary implications, where the set of all these implications forms a tree. The reasoning needed to find out if a proposition p is known at t can be done by a simple backward chaining over the tree edges that end in $p(t)$, followed by a SAT check to see if the initial state formula implies the disjunction of the reachable tree leafs.

Obviously, our relaxation induces an over-approximation of reachability. If act is a plan for a task $(A, \mathcal{I}, G)$ then, for any relaxation function $|_1^+$ for A, $act|_1^+$ is a plan for $(A|_1^+, \mathcal{I}, G)$. Note that this implies that, if there is no plan for a relaxation of a task, then there is no plan for the real task either. So if our reasoning about relaxed tasks is complete relative to the relaxation, and figures out that there is no relaxed plan to a search state, then that state can be safely excluded from the search.

4.2. Solving relaxed tasks

Our algorithms for solving relaxed conformant tasks are a natural extension to the algorithms used in FF [6], and are best presented and understood this way. We now first outline FF's computations, then we present their extension to our conformant setting/relaxation. We prove that the overall algorithm is sound and complete relative to the relaxation.

FF's method of solving relaxed tasks is a specialized version of the Graphplan algorithm [15]. Starting from a world state w, build a *relaxed planning graph* as a sequence of alternating proposition layers $P(t)$ and action layers $A(t)$, where $P(0)$ is the same as w, $A(t)$ is the set of all actions whose preconditions are contained in $P(t)$, and $P(t+1)$ is $P(t)$ plus the add effects (with fulfilled conditions) of the actions in $A(t)$. From a proposition layer $P(m)$ in which the goals are contained one can find a relaxed plan by a simple backchaining loop. One selects achieving actions at layers $t < m$ for all goals in $P(m)$, one inserts those actions' preconditions and the respective effect conditions as new subgoals (which by construction are at layers below the respective actions), then one steps backwards and selects achievers for the subgoals. The heuristic value $h(w)$ for w then is the number of actions selected in backchaining—the length of the relaxed plan. If there is no relaxed plan then the planning graph will reach a fix point $P(t) = P(t+1)$ without reaching the goals; $h(w)$ is then set to ∞ (excluding the state from the search space).

In the conformant setting, we take the sets $P(t)$ and $A(t)$ to contain the propositions that are known to hold at t and the actions that are known to be applicable at t, respectively. Additionally we introduce sets $uP(t)$ of propositions that are unknown at step t, i.e. propositions that are true when starting from some initial world states, but that are false when starting from others. We introduce machinery that decides if a proposition p is known at t. The machinery maintains a set *Imp* of implications between propositions at adjacent time steps. A variation of the machinery determines, during relaxed plan extraction, what actions must be inserted into the relaxed plan in order to make p known at t, if that is required.

The process is additionally complicated by the fact that, in difference to the classical setting where all relevant information is contained in the world state w in question, in our search space representation only *partial* knowledge about the belief state is available. All we have in memory are the sets of known and unknown propositions in the belief state. These do not tell us anything about any *constraints* between the unknown propositions that might follow from the actions we executed on the path to the state. If we ignore such constraints arising from the past, then the relaxed planning algorithm becomes incomplete. Consider, for example, a goal proposition g that can be made true

by an action a_p, with a conditional effect that achieves g given we have p (whose value is initially unknown). If we are in the belief state s where only a_p was applied, then there is a constraint $\neg p \vee g$ in our unknown propositions. Ignoring this constraint, a relaxed plan treats s exactly like the initial state, and (potentially) needs to make use of a_p although that is unnecessary in s. For one thing, this makes the resulting heuristic function lose distinctions between states, since different belief states are treated equally. Even more importantly, a predecessor action might no longer be applicable, which may make us lose the *ability* to solve the problem, even in its relaxed form (example: a_p has as precondition, and deletes, some proposition that cannot be re-achieved).

Due to these observations, we need to let the relaxed planning algorithm reason about what constraints follow from the actions in *act*. In our approach, in a similar fashion to how we compute known propositions, this reasoning is done by keeping track of the conditional effects that may occur along the execution of *act*, depending on the initial world state. The reasoning is based on the same set *Imp* of (timed) implications that we maintain to determine known propositions at later layers of the relaxed planning graph. Thus the reasoning is easiest understood as additional relaxed planning graph layers referring to the past. The layers are indexed from $-n, \ldots, -1$, corresponding to the actions in *act*, n being the length of *act*. See the conformant relaxed planning graph (*CRPG*) building algorithm in Fig. 2.

We first explain Fig. 2, then we consider an example, then we state that the CRPG algorithm is complete, then we show how one can extract a relaxed plan from a successfully built CRPG.

---

procedure build-CRPG($act, A, \mathcal{I}, G, |_1^+$),
  returns a Bool saying if there is a relaxed plan for the belief state
    given by $act = \langle a_{-n}, \ldots, a_{-1} \rangle$, and
  builds data structures from which a relaxed plan can be extracted
$Imp := \emptyset$, $P(-n) := \{p \mid p \text{ is known in } \mathcal{I}\}$, $uP(-n) := \{p \mid p \text{ is unknown in } \mathcal{I}\}$
for $t := -n \ldots -1$ **do**
  $A(t) := \{a_t|_1^+, NOOP\}$
  build-timestep(t, $A(t)$)
endfor
$t := 0$
while $G \not\subseteq P(t)$ **do**
  $A(t) := \{a|_1^+ \mid a \in A,\ pre(a) \subseteq P(t)\} \cup \{NOOP\}$
  build-timestep(t, $A(t)$)
  if $P(t+1) = P(t)$ and
    $uP(t+1) = uP(t)$ and
    $\forall p \in uP(t+1) : Impleafs(p(t+1)) = Impleafs(p(t))$ **then**
    return FALSE
  endif
  $t := t + 1$
endwhile
$m := t$, **return** TRUE

procedure build-timestep(t, A),
  builds $P(t+1)$, $uP(t+1)$, and the implication edges from t to $t+1$,
  as induced by the action set A
for all effects e of an action $a \in A$ **do**
  if $con(e) \in P(t)$ **then** $P(t+1) := P(t+1) \cup add(e)$ **endif**
  if $con(e) \in uP(t)$ **then**
    $uP(t+1) := uP(t+1) \cup add(e)$
    $Imp := Imp \cup \{(con(e)(t), p(t+1)) \mid p \in add(e)\}$
  endif
endfor
for all $p \in uP(t+1) \setminus P(t+1)$ **do**
  if $\mathcal{I} \rightarrow \bigvee_{l \in Impleafs(p(t+1))} l$ **then** $P(t+1) := P(t+1) \cup \{p\}$ **endif**
endfor
$uP(t+1) \setminus = P(t+1)$

Fig. 2. Building a conformant relaxed planning graph (CRPG).

Let us just read Fig. 2 from top to bottom. The CRPG is initialized by setting the implication set *Imp* to ∅, and setting the initial proposition layer $P(-n)$ and $uP(-n)$ to the information provided by $\mathcal{I}$ (we have this information from our reasoning about known propositions). The **for** loop then builds the time steps $-n, \ldots, -1$ with action layers corresponding to the actions in *act*, relaxed with $|_1^+$. We additionally introduce a *NOOP* action, to simplify the presentation. The *NOOP* simply transports all propositions from layer t to layer $t+1$. More formally, $pre(NOOP) = \emptyset$ and

$$E(NOOP) = \big\{(\{p\}, \{p\}, \emptyset) \mid p \text{ is a proposition in the task at hand}\big\}.$$

The **while** loop in Fig. 2 builds time steps t from 0 on until a proposition layer m is reached that contains the goals G. The actions in each layer t are those whose preconditions are reached at t (plus the *NOOP*), relaxed with $|_1^+$. The algorithm terminates with return value FALSE if a termination criterion is met. We focus on the termination criterion further below. Right now let us consider how the time steps are built. The first **for** loop proceeds over all effects of the actions in the given set. Effects whose condition is known to hold at t yield (potentially) new known propositions at $t+1$. Effects whose condition can hold depending on the initial world state yield (potentially) new such propositions at $t+1$, together with the appropriate implication edges in *Imp*. Note that, to ease the presentation, we assume here that there are no unconditional effects, i.e. no effects e for which $con(e) = \emptyset$. This is no restriction: for such effects, one can simply introduce a new proposition p that is always true, and set $con(e) := \{p\}$. With this we can assume that all effect conditions (of the relaxed actions) contain *exactly* one proposition. To emphasize this, in the notation we use the set $con(e)$ as if it was a single proposition. The second **for** loop of the build-timestep procedure goes over all p that may be true at $t+1$, but that were not yet proved to always be true at $t+1$. It is checked if the initial state formula implies the disjunction of the leafs of the *Imp* tree that are reachable from $p(t+1)$: $Impleafs(p(t'))$, for any t', is defined as

$$Impleafs(p(t')) := \big\{l \mid \text{Exists a path in } Imp \text{ from } l(-n) \text{ to } p(t')\big\}.$$

In our implementation, such leaf sets are computed by a simple backchaining loop over the edges ending in $p(t')$. The implication $\mathcal{I} \rightarrow \bigvee_{l \in Impleafs(p(t+1))} l$ is checked by a call to the SAT solver, and if the implication holds, p is inserted into the known propositions at $t+1$. As a last step, the procedure removes from the set of unknown propositions, i.e., from those propositions for which it has been shown that there is at least one initial world state that makes them become true, all p that were proved to be always true (i.e., known) at $t+1$.

To illustrate the CRPG algorithm, let us reconsider the example from Section 3.

Example 2. A robot is initially at one out of two locations, modeled as $\mathcal{I} = \{at\text{-}L_1 \vee at\text{-}L_2, \neg at\text{-}L_1 \vee \neg at\text{-}L_2\}$. Our goal is to be at L_2, and we have a *move-right* action that has an empty precondition, and the conditional effect $(\{at\text{-}L_1\}, \{at\text{-}L_2\}, \{at\text{-}L_1\})$. The CRPG to the initial state—i.e. the CRPG to the empty sequence $act = \langle\rangle$—is computed as follows. $P(0)$ is set to ∅, $uP(0)$ is set to $\{at\text{-}L_1, at\text{-}L_2\}$. In the first iteration, $t = 0$, of the **while** loop, build-timestep is called with $A(0) = \{move\text{-}right, NOOP\}$. In the first **for** loop of build-timestep, the *move-right* action yields the implication edge $(at\text{-}L_1(0), at\text{-}L_2(1))$, while the *NOOP* yields the edges $(at\text{-}L_1(0), at\text{-}L_1(1))$ and $(at\text{-}L_2(0), at\text{-}L_2(1))$. In the second **for** loop, we find that $Impleafs(at\text{-}L_1(1)) = \{at\text{-}L_1(0)\}$ which is not implied by $\mathcal{I}$; however, $Impleafs(at\text{-}L_2(1)) = \{at\text{-}L_1(0), at\text{-}L_2(0)\}$ whose disjunction *is* obviously implied by $\mathcal{I}$. Consequently, $at\text{-}L_2$ is moved into $P(1)$ and we end up with $P(1) = \{at\text{-}L_2\}$, $uP(1) = \{at\text{-}L_1\}$. So in the next iteration, $t = 1$, of the **while** loop, the algorithm terminates with success and returns TRUE. Now, say we call the CRPG algorithm on the belief state given by the action sequence $act = \langle move\text{-}right\rangle$. In this case the whole above process is moved one step "to the left". We obtain $P(-1) = \emptyset$, and $uP(-1) = \{at\text{-}L_1, at\text{-}L_2\}$. $A(-1)$ is set to $\{move\text{-}right, NOOP\}$, which leads to $P(0) = \{at\text{-}L_2\}$ and $uP(0) = \{at\text{-}L_1\}$ just like above. As we will explain below, actions for inclusion in the relaxed plan to the state are only selected for goals at layers $t > 0$, so with the goal $at\text{-}L_2$ being contained in $P(0)$ our relaxed plan will be empty. (In difference to the initial state, where $at\text{-}L_2$ is first contained in $P(1)$ and will lead to the selection of *move-right* into the relaxed plan.)

The reasoning described above is adequate for our relaxation (more precisely, Lemma 1 in Appendix A holds) because *Imp* captures all dependencies introduced by the relaxed actions between propositions at time steps. Note that the *NOOP* action introduces implication edges of the form $(p(t), p(t+1))$ saying that a proposition p will still be true at $t+1$ if it was true at t (remember that there are no delete lists). The termination criterion of the CRPG

computation asks if, from time step t to time step $t+1$, neither the known propositions nor the unknown propositions nor the reachable *Imp* tree leafs of the latter propositions have changed. It is relatively easy to see that, in this case, the same would hold true at all future time steps $t' > t$, implying that the goals will never be reached.

More formally, the following holds.

Theorem 2 *(CRPG-completeness). Given a conformant task $(A, \mathcal{I}, G)$, an executable action sequence act, and a relaxation function $|_1^+$ for A. If* build-CRPG$(act, A, \mathcal{I}, G, |_1^+)$ *returns FALSE then there is no relaxed plan for $(A, \mathcal{I}, G)$ that starts with the sequence act.*

Proof sketch. The proof is based on two observations. First, the propositions in the sets $P(t)$ (respectively $uP(t)$) are exactly those propositions that are known (respectively unknown) after executing all the actions in the action layers up to $A(t-1)$. Second, if the CRPG algorithm returns FALSE in some iteration then the termination criterion would also hold in all future iterations. With the first observation, if there is an m-step relaxed plan for $(A, \mathcal{I}, G)$ that starts with *act*, then the goals are contained in $P(m)$. With the second observation, if the CRPG algorithm returns FALSE then no such layer m exists. □

If the CRPG building algorithm succeeds in reaching the goals, then a relaxed plan is extracted and the length of that relaxed plan is used to generate the heuristic value of the belief state—see below. Relaxed plan extraction proceeds as specified in Fig. 3.

We explain Fig. 3, then we reconsider the example, then we state that the CRPG algorithm is sound, then we explain how the heuristic value to the state is generated.

The process makes use of proposition sets $G(1), \ldots, G(m)$, which are used to store the goals and sub-goals arising at layers $1 \leqslant t \leqslant m$ during relaxed plan extraction. The sub-goal procedure simply inserts all given propositions as sub-goals at their first layer of appearance in the CRPG. The sub-goals are considered layer-by-layer, starting from

```
procedure extract-CRPlan(CRPG(act, A, I, G, |_1^+), G),
          selects actions from A(0), ..., A(m − 1) that
          form a plan for (A|_1^+, I, G) when appended to act|_1^+
sub-goal(G)
for t := m, ..., 1 do
  for all g ∈ G(t) do
    if ∃a ∈ A(t − 1), e ∈ E(a), con(e) ∈ P(t − 1), g ∈ add(e) then
      select one such a /* and e */ at t − 1
      sub-goal(pre(a) ∪ con(e))
    else
      let L be a minimal subset of Impleafs(g(t)) s.t. I → ⋁_{l∈L} l
      for all i ⩾ 0, actions a ∈ A(i) \ {NOOP}, and effects e ∈ E(a) s.t.
          e is responsible for an edge in a path from l(−n), l ∈ L, to g(t) do
          select a at i
          sub-goal(pre(a))
      endfor
    endif
  endfor
endfor

procedure sub-goal(P),
          inserts the propositions in P as sub-goals
          at the layers of their first appearance in the CRPG
for all p ∈ P do
  let t_0 be the smallest t s.t. p ∈ P(t)
  if t_0 ⩾ 1 then G(t_0) := G(t_0) ∪ {p} endif
endfor
```

Fig. 3. Extracting a conformant relaxed plan.

the top layer m going downwards to layer 1. For each sub-goal g at layer t, supporting actions are selected into the relaxed plan. If there is an action a at layer $t-1$ that guarantees to always achieve g, then a is selected at $t-1$. Otherwise, a minimal subset L of $\mathit{Impleafs}(g(t))$ is determined such that $\mathcal{I} \rightarrow \bigvee_{l \in L} l$. This is done by starting from $L := \mathit{Impleafs}(g(t))$ and, iteratively, removing propositions p from L if $\bigvee_{p \neq l \in L} l$ is still implied by $\mathcal{I}$, until no such p is left in L. Then all actions are selected, at the respective times, that are responsible for the implication paths from L at $-n$ to g at t. The motivation behind the use of a minimal implied subset L is that we want to avoid the inclusion of superfluous actions into the relaxed plan. Such superfluous actions lead to an unnecessary over-estimation of the real goal distance, and can seriously affect the informativity of the resulting heuristic function.

Selection of actions is to be understood as a set union operation, i.e. if an action a is selected that was selected at the same time before, then nothing happens. Note that, in the first case of the **if** statement, when an action guarantees to always achieve g, then that action cannot be the *NOOP* because otherwise g would be contained in $P(t-1)$ already.

We reconsider the illustrative example above.

Example 3. A robot is initially at one out of two locations, $\mathcal{I} = \{\mathit{at}\text{-}L_1 \vee \mathit{at}\text{-}L_2, \neg \mathit{at}\text{-}L_1 \vee \neg \mathit{at}\text{-}L_2\}$. The goal is to be at L_2, and the *move-right* action has an empty precondition, and the conditional effect $(\{\mathit{at}\text{-}L_1\}, \{\mathit{at}\text{-}L_2\}, \{\mathit{at}\text{-}L_1\})$. To the initial state, we obtain the CRPG $P(0) = \emptyset, uP(0) = \{\mathit{at}\text{-}L_1, \mathit{at}\text{-}L_2\}$, $P(1) = \{\mathit{at}\text{-}L_2\}, uP(1) = \{\mathit{at}\text{-}L_1\}$, and $\mathit{Imp} = \{(\mathit{at}\text{-}L_1(0), \mathit{at}\text{-}L_2(1)), (\mathit{at}\text{-}L_1(0), \mathit{at}\text{-}L_1(1)), (\mathit{at}\text{-}L_2(0), \mathit{at}\text{-}L_2(1))\}$. Relaxed plan extraction starts from layer $m = 1$, with $G(1) = \{\mathit{at}\text{-}L_2\}$. There is no action that guarantees achievement of $\mathit{at}\text{-}L_2$, and we get $L = \mathit{Impleafs}(\mathit{at}\text{-}L_2(1)) = \{\mathit{at}\text{-}L_1(0), \mathit{at}\text{-}L_2(0)\}$. The edges from L at 0 to $\mathit{at}\text{-}L_2$ at 1 are $(\mathit{at}\text{-}L_1(0), \mathit{at}\text{-}L_2(1))$ and $(\mathit{at}\text{-}L_2(0), \mathit{at}\text{-}L_2(1))$. The former edge is induced by *move-right* at 0, the latter edge is induced by *NOOP* at 0. So (only) *move-right* at 0 is selected into the relaxed plan. In the CRPG built to $\mathit{act} = \langle \mathit{move\text{-}right} \rangle$, as explained above we have $\mathit{at}\text{-}L_2 \in P(0)$. So we get $m = 0$ and relaxed plan extraction does not perform any sub-goal achievement iterations, thus delivering an empty relaxed plan.

It is relatively easy to see that the actions selected by extract-CRPlan form a relaxed plan for the belief state given by act.

Theorem 3 *(CRPG-soundness). Given a conformant task $(A, \mathcal{I}, G)$, an executable action sequence act, and a relaxation function $|_1^+$ for A, s.t.* build-CRPG$(\mathit{act}, A, \mathcal{I}, G, |_1^+)$ *returns TRUE. Let $A(0)^s, \ldots, A(m-1)^s$ be the actions selected from $A(0), \ldots, A(m-1)$ by* extract-CRPlan$(\mathit{CRPG}(\mathit{act}, A, \mathcal{I}, G, |_1^+), G)$. *Then $\mathit{act}|_1^+$ concatenated with $A(0)^s, \ldots, A(m-1)^s$ in an arbitrary linearization is a plan for $(A|_1^+, \mathcal{I}, G)$.*

Proof sketch. It can be shown by induction over t that, for any proposition g that is made a sub-goal at time t $(0 < t \leqslant m)$ during the execution of extract-CRPlan, g is known after executing the actions in $\mathit{act}|_1^+$ and $A(0)^s, \ldots, A(t-1)^s$. This shows the claim. The main argument in the induction is that, if actions were selected for all effects that were responsible for an edge in a path from $l(-n)$, $l \in L$, to $g(t)$, where $\mathcal{I} \rightarrow \bigvee_{l \in L} l$, then in every $I \in 2^{\mathcal{I}}$ one $l \in L$ is true, and the actions on the path from $l(-n)$ to $g(t)$ make g true at t. □

Theorems 2 and 3 together imply that build-CRPG$(\mathit{act}, A, \mathcal{I}, G, |_1^+)$ returns TRUE if and only if there is a relaxed plan for $(A, \mathcal{I}, G)$ that starts with act.

If the CRPG returns FALSE, we set our heuristic value h to ∞: the state is proved to be a *dead end*, i.e. a state from which the goals can't be reached (this holds because, as pointed out above, $|_1^+$ induces an over-approximation of reachability). The state will be excluded from the search space. If the CRPG succeeds in reaching the goals, let h be the number of actions selected by extract-CRPlan$(\mathit{CRPG}(\mathit{act}, A, \mathcal{I}, G, |_1^+), G)$, i.e., the number of actions in the relaxed plan to the belief state. As in the classical setting of FF, our intention is to use h as the heuristic value for the state. There is, however, one subtlety here in the conformant setting. We relax the semantics of not only the actions in the CRPG layers at times $0, \ldots, m$, but also of the actions at times $-n, \ldots, -1$, i.e., of *the actions on the path to the state*. Because of this, the CRPG is an over-approximation not only of what we will be able to achieve in the future, but also of what we already achieved in the past. In consequence, it may be that we obtain an empty relaxed plan—the goals are all reached at $P(0)$ already—although the goals are *not* known in the state. So our heuristic value may be 0 in non-goal states. Since normally a heuristic value of 0 is taken to indicate goal states, we add 1 to h in case the goals are not known after *act* (which we know about from our reasoning about the search states). Note that the

phenomenon—over-approximation of what was achieved in the past—may seriously affect the informativity of the heuristic function in regions of the search space where many actions were applied already. We explain below, amongst other things, a variation of the heuristic function with which we have tried to overcome this problem.

Altogether, the above processes define a heuristic h as a function from belief states into $\mathbf{N}_0 \cup \{\infty\}$, i.e. into the natural numbers with 0 and $\{\infty\}$. States s with $h(s) = 0$ are goal states, states with $h(s) = \infty$ are proved unsolvable and can be skipped.

4.3. Optimizations

One can spare some time in the creation of the CRPG, and obtain a somewhat more informative heuristic function, by re-using the knowledge stored along the search path to the state, i.e., along the action sequence *act*, when building the CRPG layers $-n, \ldots, -1$. Instead of computing the proposition layers $P(-n), uP(-n), \ldots, P(-1), uP(-1)$ by the same machinery as used for the higher CRPG layers, one can simply set these proposition layers to the values stored in the search states along *act*, as were computed during the search. The *Imp* edges can be inserted accordingly. This obviously saves some time, and also it ameliorates the problem outlined above, over-estimation of the past achievements. Note that, however, the reasoning done on the *Imp* tree still over-estimates the semantics of the actions in *act*. One of the variations described below avoids this problem altogether, at the cost of more expensive SAT reasoning.

There is no need to decide off-line, for each effect, exactly which of the condition propositions one ignores. Indeed, in some of the computations made when building the CRPG, it is not necessary to ignore all but one of the effect conditions. One can avoid applying effects that are known to not occur, and one can avoid ignoring the known aspects of an effect condition. In our implementation, say we build a time step t and, during the first **for** loop of build-timestep, consider an action effect e. We look at the original form of e, with the whole effect condition. If $con(e) \subseteq P(t)$, the effect condition is known to hold and all $p \in add(e)$ are inserted into $P(t+1)$. If $con(e) \not\subseteq P(t) \cup uP(t)$, the effect is known to not occur, and we skip it. If the above two cases do not hold, i.e. we have $con(e) \not\subseteq P(t)$ and $con(e) \subseteq P(t) \cup uP(t)$, then the effect *possibly* occurs. We then (arbitrarily) select one $c \in con(e)$ such that $c \in uP(t)$, and insert all $p \in add(e)$ into $uP(t+1)$, with edges $(c(t), p(t+1))$.

4.4. Variations

Throughout the rest of the article, we refer to the h values delivered by the above machinery, including the optimizations, as the *initial-formula* heuristic function, denoted with $h^{\mathcal{I}}$ to indicate that it is obtained by SAT checks against $\mathcal{I}$. We also call $h^{\mathcal{I}}$ the *standard* heuristic function because we use it as the default in our implemented system.

In the first variation of the heuristic, called the *incomplete* heuristic function and denoted as h^{inc}, we do not do any SAT calls to check the implications $\mathcal{I} \rightarrow \bigvee_{l \in L} l$. We only do a simple sufficient test. The motivating observation is that, often, in our examples, when $\mathcal{I}$ implied $\bigvee_{l \in Impleafs(p(t))} l$, the reason was simply a clause in $\mathcal{I}$ that was a subset of $Impleafs(p(t))$. (Note that this is also the case for $Impleafs(at\text{-}L_2(1)) = \{at\text{-}L_1(0), at\text{-}L_2(0)\}$ in our illustrative example.) So our sufficient condition only sees if there is a clause C in $\mathcal{I}$ such that $C \subseteq L$. If that is not the case, then we assume that the implication does not hold. This can, obviously, help to save runtime. Likewise obviously, the resulting CRPG is no longer complete relative to the relaxation, i.e., the CRPG can return FALSE even if there is a relaxed plan. So, if the CRPG cannot reach the goals, to preserve the overall completeness of the search algorithm, h^{inc} returns some large integer value instead of ∞.

Our second variation of $h^{\mathcal{I}}$ is called the *state-formula* heuristic function, and denoted with $h^{\phi(act)}$. Recall that $h^{\mathcal{I}}$ relaxes the semantics of all actions including those in *act*, and checks *Imp* leaf disjunctions for implication by $\mathcal{I}$. In contrast, the relaxed planning algorithm behind $h^{\phi(act)}$ only relaxes the semantics of the actions in the "future" layers of the CRPG, $t \geqslant 0$, and checks *Imp* leaf disjunctions for implication by *the formula* $\phi(act)$ *that encodes the semantics of the belief state given by act*. Indeed, the CRPG now only contains layers t with $t \geqslant 0$. Precisely, the variation is obtained from Fig. 2 as follows. Skip the first **for** loop. Set $P(0)/uP(0)$ to the known/unknown propositions after *act*, and start with the **while** loop. Replace, in build-timestep, the condition $\mathcal{I} \rightarrow \bigvee_{l \in Impleafs(p(t+1))} l$ with $\phi(act) \rightarrow \bigvee_{l \in Impleafs(p(t+1))} l(n)$. Otherwise, the algorithm remains unchanged. In the relaxed plan extraction algorithm, Fig. 3, the only change made is that now L is defined to be a minimal subset of $Impleafs(g(t))$ such that $\phi(act)$, rather than $\mathcal{I}$, implies $\bigvee_{l \in L} l$.

The $h^{\phi(act)}$ approach has the advantage that reasoning about the past is precise, with no over-estimation of what was achieved by *act*. In particular, if the returned relaxed plan is empty, then this means that s is a goal state. The disadvantage of $h^{\phi(act)}$ is that checking implication against the formulas $\phi(act)$ is typically much more costly than checking implication against $\mathcal{I}$. The latter formula is the same for all search states. The former formula grows with the length of *act*, which are up to a few hundred steps in some of our examples. Note that the checks for $\phi(act) \rightarrow \bigvee_{l \in Impleafs(p(t+1))} l(n)$ need to be done every time when the CRPG has to test if an unknown proposition becomes known at a layer, and every time the relaxed plan selects such a proposition as a sub-goal. (In the latter case, even several implication checks are needed in order to identify a minimal implied subset.)

We remark that, in our experiments, often $h^{\phi(act)}$ and h^{inc} yielded performance very similar to that of $h^{\mathcal{I}}$. Details are in Section 6.

5. Repeated states

Forward search spaces in planning are, generally, graphs, i.e., the same search states can be reached via different search paths. In many examples, this happens so frequently that checking for repeated search states becomes crucial for performance. (For example, repeated search states occur when several actions are independent of each other and can be applied in any order.)

We avoid repeated search states by a hash table technique combined with a belief state equivalence test (or a belief state domination test, see below). When we perform search, we check via a hash table lookup whether the new belief state is equivalent to (dominated by) some previous belief state in the same hash entry. If so, the new belief state is pruned. The hash value of a state is a function of the union of the propositions that are known or unknown in the state, which ensures that equivalent belief states end up in the same hash entry (state dominations may be missed due to different hash values).

In the classical setting, testing equality of world states is straightforward and easy: the states are proposition sets. In the conformant setting, one must test if two belief states, i.e., two *sets* of world states, are equal. We did not find a way to do this based on our sparse representation of belief states. Instead, we test a stronger, sufficient but not necessary, criterion which we call belief state *equivalence*. Let s and s' be two belief states reached via the action sequences *act* and *act*$'$, respectively. We say that s is equivalent to s' iff for every possible initial world state $I \in 2^{\mathcal{I}}$ the world state upon execution of *act* in I is equal to the world state upon execution of *act*$'$ in I. Note that s and s' may be equal but not equivalent because the "ordering" of their world states—the correspondence to the initial world states—may differ even though, as sets of world states, s and s' are equal. However, if s and s' are equivalent then obviously they are also equal.

The equivalence relation can be tested based on satisfiability checks very similar to those that we do for computing the known propositions. The key observation is the following. Two belief states s and s' are equivalent iff for all propositions p there is no $I \in 2^{\mathcal{I}}$ starting from which p has different truth values in (the respective world states in) s and s'. From left to right, if s and s' are equivalent then obviously all p show the same behavior from all I. From right to left, if there was an I that yielded different world states w and w' in s and s', then at least one p would have different values in w and w', in contradiction.

With the above, we can use the following satisfiability tests to decide about belief state equivalence. Let $\phi_{I=}(act)$ and $\phi_{I=}(act')$ be the formulas constructed for the action sequences *act* and *act*$'$, such that $\phi_{I=}(act)$ and $\phi_{I=}(act')$ share the same propositional variables for time 0 propositions but have different propositional variables for all times greater than 0 (i.e., the conjunction of these formulas captures the semantics of executing *act* and *act*$'$ in the *same* initial world state). For a proposition p, let $p(act)$ and $p(act')$ denote p's value (the respective propositional variable) following *act* and *act*$'$, respectively. Then s is equivalent to s' iff the formulas $\phi_{I=}(act) \wedge \phi_{I=}(act') \wedge p(act) \wedge \neg p(act')$ and $\phi_{I=}(act) \wedge \phi_{I=}(act') \wedge \neg p(act) \wedge p(act')$ are unsatisfiable for all p.[10]

One can, of course, optimize the equivalence test by making use of the knowledge that was already computed for s and s'. State equivalence can only hold if s and s' agree on the sets of known/negatively known/unknown

[10] Note that this set of SAT tests is basically a decomposed form of the non-CNF entailment test $\phi_{I=}(act) \wedge \phi_{I=}(act') \wedge \neg \bigwedge_p [(\neg p(act) \vee p(act')) \wedge (p(act) \vee \neg p(act'))]$, i.e., the test that asks if all p behave equivalently at the ends of *act* and *act*$'$. The decomposition consists of putting $\neg \bigwedge_p [(\neg p(act) \vee p(act')) \wedge (p(act) \vee \neg p(act'))]$ into negation normal form, and proving unsatisfiability for each individual member of the resulting DNF.

propositions—which, in our code, is equivalent to s and s' falling into the same entry of the hash table. The SAT calls need only be made for those p that are unknown (in both s and s').

In many domains, including all but one of our experimental domains, a stronger form of repeated state pruning is valid. In these domains it is the case that it can only be better to have more propositions true. More formally, we say that a world state w *dominates* a world state w' if $w \supseteq w'$. In many domains, if w dominates w', then every action sequence that solves the goal starting from w' also solves the goal when starting from w. Then, if in a classical search one has already seen w, and w' comes up as a new state, w' can be skipped without losing completeness. In the conformant setting, we say that a belief state s dominates a belief state s' if, for every initial world state $I \in 2^{\mathcal{I}}$, the corresponding world state in s (the state after executing act) dominates the corresponding world state in s' (the state after executing act'). If it is always better to have more propositions true, and s dominates s', then every conformant plan from s' is also a conformant plan from s. So if one has already seen s then s' can be skipped. Belief state domination can be tested by the exact same technique as explained for the equivalence test above, except that one needs only a single SAT call per proposition, namely that for $\phi_{I=}(act) \wedge \phi_{I=}(act') \wedge \neg p(act) \wedge p(act')$. It is easy to see that s dominates s' iff this formula is unsatisfiable for all p.[11] Obviously, one can skip some SAT calls by observing that s can only dominate s' if its known (known or unknown) propositions are a superset of those known (unknown) in s'. SAT calls need only be made for those p unknown in both s and s'.

In comparison to the equivalence test, the domination test has more pruning potential on top of being cheaper in terms of the number of SAT calls made. But in our conformant planning setting, it is not generally the case that it can only be better to have more propositions true. The reason lies in the conditional effects (in pure STRIPS, obviously having more propositions true can only be better). If $w \supseteq w'$, and $p \in w \setminus w'$, then there may be a conditional effect that has p among its conditions, and that deletes important propositions (that would not be deleted when executing the same action in w'). However, there is a simple sufficient condition implying that the latter cannot happen. Say a conditional effect e is of the form $(\{c\}, add, (del \cap SREL) \subseteq \{c\})$. Here $SREL$ denotes the set of all propositions that occur either in the goal or in some action precondition or in some effect condition. That is, e has only a single condition c and a delete list whose only delete that may be needed is c. Then if e occurs from w but not from w', the only important proposition that may be deleted is an "additional" one, i.e. one in $w \setminus w'$. With this, it is easy to see that, if all effects are of the form $(\{c\}, add, (del \cap SREL) \subseteq \{c\})$, then with $w \supseteq w'$ every plan from w' is also a plan from w. It is then valid to prune a belief state s' if one has already seen a belief state s that dominates s'.

In our implementation of Conformant-FF, before search starts we (automatically) check if all conditional effects have the form explained above. If so, the repeated states checking uses the stronger belief state domination test. Otherwise, the full equivalence test is used. In all our experimental domains, with a single exception (the Omelette domain, namely), all conditional effects have the necessary form, and the domination test is used.

In a few cases in our experiments we found that repeated states checking incurred a huge overhead, rendering instances prohibitively hard that were otherwise—when turning the repeated states check off—easy to solve. This phenomenon occurs in search spaces where no or only very few propositions become known/negatively known before a plan is found. In such a situation, almost every visited state ends up in the same hash entry, since the hashing function has no means to distinguish between them. Every time a new state is generated, it is compared to (almost) every state visited before. So, in total, the number of comparisons made is a square function in the number of explored search states. Every single comparison can involve several SAT calls, and so, clearly, checking for repeated states becomes all-consuming.

To overcome the above deficiency, we have also implemented a weaker variant of repeated states checking, that we call *checking for stagnating states*. There, a new search state s' is only compared to the states s visited *on the path to* s'. The total number of state comparisons is then bounded by the number of explored states multiplied with the depth of the search tree (rather than the size of that tree, as before). The drawback is, clearly, a loss of pruning potential.

The default setting in our implementation is a full repeated states check. Most of the time, at least as reflected by our experiments, this behaves better than, or equally good as, the limited stagnation check. An alternative way to overcome the potential weakness of full repeated states checking may be to use fast pre-checks to see if the SAT calls can be avoided. We discuss this in the outlook in Section 9.

[11] Similarly to above, this set of SAT calls is a decomposed version of the test $\phi_{I=}(act) \wedge \phi_{I=}(act') \wedge \neg \bigwedge_p [p(act) \vee \neg p(act')]$.

6. Results

We implemented the techniques described in the previous sections in C. We evaluated the resulting system, Conformant-FF, on a collection of benchmarks including traditional conformant benchmarks, and classical benchmarks enriched with uncertainty. We compared Conformant-FF's performance in detail to that of MBP and KACMBP. We also ran some comparative tests with GPT and with POND.

In the next subsection, we give some implementation details of Conformant-FF, thereafter a subsection gives details on our experimental setup. Then two subsections give our results for Conformant-FF, MBP, and KACMBP. The first of these subsections considers the traditional conformant benchmarks, the second one considers the enriched classical benchmarks. A final subsection describes the results we obtained with GPT and POND.

6.1. Implementation

We implemented the Conformant-FF system in C, starting from the FF code.[12] Conformant-FF's overall architecture is identical to that of FF. An outline of the architecture is this:

(1) Do one trial of "enforced hill-climbing": set the current search state s to the initial state; while s is not a goal state:
   (a) Starting from s, perform a breadth-first search for a world state s' such that $h(s') < h(s)$. During search: avoid repeated states; cut out states s' with $h(s') = \infty$; and expand only the successors of s' generated by the actions in $H(s')$.
   (b) If no s' with $h(s') < h(s)$ has been found then fail, else $s := s'$.
(2) If enforced hill-climbing failed, invoke complete best-first search, i.e., starting from the initial state expand all world states in order of increasing h value, avoiding repeated states.

Here, $h(s)$ denotes the heuristic value of a search state—the number of actions in a relaxed plan to the state. $H(s)$ denotes the so-called *helpful actions*, which are the only actions considered by the planner during hill-climbing. The helpful actions to a search state are those that could be selected to *start* the relaxed plan. In the classical setting these are those actions at the lowest level of the relaxed planning graph that add a subgoal (at this level). In the conformant setting, $H(s)$ are the actions at the lowest level of the relaxed planning graph that add a subgoal, or that were selected for an implication path during relaxed plan extraction.

Conformant-FF uses a largely naive standard DPLL solver for the CNF reasoning. We also implemented a version using Chaff [16] as the SAT solver. But typically, when running Conformant-FF thousands of CNF formulas have to be considered, each of which is very easy to solve—most of the time, in our experiments, a single run of unit propagation sufficed to find a satisfying assignment or to prove unsatisfiability. In the version using Chaff, communicating all the CNFs to Chaff, and the time taken by Chaff to build its (rather complex) internal data structures, quickly became prohibitive for scaling while the overall time taken to actually solve the CNFs was negligible. So we implemented our own DPLL-based SAT solver within Conformant-FF, working directly on Conformant-FF's internal data structures. The implementation is a naive standard backtracking realized by recursive function calls, using no clever data structures whatsoever except, for each literal, a linked list to the clauses it participates in. The latter lists are used to quickly access the relevant clauses when a literal gets set by unit propagation. We did not implement watched literals because in our experiments typically around 95% of the generated clauses were binary anyway.[13] We did not even implement a variable selection heuristic (i.e., the variables are selected in an arbitrary order) because, as said above, most of the time a single run of unit propagation sufficed. For the same reason, we did not bother at all with clause learning techniques. If not run on examples with much more complex formulas as the initial states, it seems unlikely that any

[12] Available at http://www.mpi-sb.mpg.de/~hoffmann/ff.html.

[13] Note that this is just a phenomenon of the benchmarks used in the experiments; in general, of course the clauses may be arbitrarily large. Our design choice here, or rather our laziness in the implementation, is dependent on practical experience, i.e., on a certain (seemingly typical) range of examples. In particular, the DPLL procedure can *not* be replaced with a 2-CNF reasoner. A promising option may be to employ 2-CNF simplification algorithms, such as e.g. [17].

more fancy SAT solving techniques than what's implemented will yield significant performance improvements for Conformant-FF.

6.2. Experimental setup

The experiments were run on a PC running at 1.2 GHz with 1 GB main memory and 1024 KB cache running Linux, with a runtime cutoff of 1200 seconds. The traditional conformant benchmark domains we used were *Bomb-in-the-toilet* (short *Bomb*), *Cube*, *Omelette*, *Ring*, and *Safe* (see also, e.g., [2] and [18]). The enriched classical benchmark domains were *Blocksworld*, *Logistics*, and *Grid*. The individual domains are described below, when we discuss the respective results. For each example instance we provide, for each planner in the experiment, the runtime taken and the length of the found plan; in the case of Conformant-FF, we also provide the number of evaluated states, i.e. the number of states for which a heuristic value was computed. We summarize the results in the text, and, where appropriate and possible, give intuitive explanations of the observed behavior.

We performed a detailed comparison of Conformant-FF with the most recent available versions of MBP and KACMBP. These two systems have been shown to be highly competitive, far more efficient than all other existing systems in many cases. E.g., they are known to outperform GPT in the traditional conformant benchmarks. (KACMBP is a more advanced version of the conformant planner in MBP—see Section 8 for more details.) We run MBP with its default settings, and we run KACMBP with the heuristic forward-search option (which seems to give the overall best performance). These settings are also used in the experimental setups included with the respective downloads.

We also performed comparisons of Conformant-FF with GPT, and with POND. The comparison to GPT was done on fewer examples but suffices to give a relatively good impression of the performance differences. The comparison to POND was done on the same set of examples as those for MBP and KACMBP, but POND often behaved very badly and it is not completely clear what the reason for that is. More details on GPT and POND are below in Section 6.5.

Conformant-FF is given as input a PDDL-like file describing the domain and the task, with obvious modifications for describing uncertainty about the initial state. Conformant-FF then generates the set of ground actions. MBP is given as input an NPDDL file, which is an extension of PDDL allowing for uncertainty, non-deterministic effects, and a rich set of goal conditions. MBP translates this input into a set of ground actions, as well. Its translation process is naive, and therefore, we made a serious effort to use various NPDDL options that reduce this set of actions. In particular, NPDDL allows for the definition of functions (which allow efficient encoding of multi-valued variables)—a capability that Conformant-FF does not have—and we tried to use this option as much as we could. In the input language of KACMBP, one explicitly specifies each ground action, also with the possibility of using multi-valued variables, which we used as much as possible.

In the experiments, we found that the performance difference between the heuristics $h^{\mathcal{I}}$ and h^{inc} was, most of the time, negligible. Thus, except in one results table where we consider exceptionally large instances, these tables contain the results for $h^{\mathcal{I}}$ and $h^{\phi(act)}$ only. In a single other case (test suite), there was a remarkable difference between the behavior of $h^{\mathcal{I}}$ and h^{inc}, which is described in the text. In a similar fashion, we report detailed results only for the default setting of repeated states checking, i.e. for the full test. Checking (only) for stagnating states yields improvements in just a few cases, which are discussed in the text.

6.3. Conformant benchmarks

Table 1 provides our experimental results in the Bomb domain. In our variant of this domain, there are $b \geqslant 1$ bombs, each of which is initially potentially armed (i.e., it is unknown if the bomb is armed or not). There are $t \geqslant 1$ toilets. One can "dunk" a bomb into a toilet. The precondition of this is that the toilet is not clogged. As effects, the action disarms the bomb if it is armed, and the toilet gets clogged. One can "flush" a toilet, with no precondition, making it un-clogged. In Table 1, the instances "Bomb-b*b*-t*t*" are the examples with b bombs and t toilets, respectively.

Regarding runtime, we observe the following. We first consider only the default version of Conformant-FF, i.e. the $h^{\mathcal{I}}$ heuristic. MBP and KACMBP are superior to Conformant-FF when there are few toilets. In the extreme case, when there is just a single toilet as in the topmost part of Table 1, Conformant-FF gets outperformed quickly as the number of bombs increases. Then, as the number of toilets rises to 5 and 10 in the next two parts of the table, we observe that Conformant-FF becomes faster while MBP and KACMBP become slower. For MBP, the loss of efficiency is dramatic,

Table 1
Results in the Bomb domain

Instance	Conformant-FF		MBP	KACMBP
	$h^{\mathcal{I}}$	$h^{\phi(act)}$		
	$t/\|S\|/l$	$t/\|S\|/l$	t/l	t/l
Bomb-b5-t1	0.01/19/9	0.00/19/9	0.01/9	0.01/10
Bomb-b10-t1	0.03/64/19	0.03/64/19	0.05/19	0.03/20
Bomb-b20-t1	0.25/229/39	0.53/229/39	0.28/39	0.07/40
Bomb-b50-t1	5.07/1324/99	10.98/1324/99	2.39/99	0.86/100
Bomb-b100-t1	129.70/5149/199	–	20.09/199	2.65/200
Bomb-b10-t5	0.02/30/15	0.00/30/15	0.74/15	0.14/20
Bomb-b20-t5	0.17/155/35	0.32/155/35	2.63/35	0.38/40
Bomb-b50-t5	4.67/1130/95	16.98/1130/95	31.43/95	1.37/100
Bomb-b100-t5	120.99/4755/195	–	275.74/195	4.38/200
Bomb-b5-t10	0.00/5/5	0.00/5/5	1.03/5	0.14/10
Bomb-b20-t10	0.05/85/30	0.04/85/30	9.64/30	0.77/40
Bomb-b50-t10	3.14/910/90	12.98/910/90	115.81/90	2.28/100
Bomb-b100-t10	109.65/4285/190	–	952.09/190	8.69/200
Bomb-b5-b5	0.00/5/5	0.01/5/5	0.20/5	0.13/10
Bomb-b10-t10	0.00/10/10	0.01/10/10	2.32/10	0.36/20
Bomb-b20-t20	0.03/20/20	0.03/20/20	46.12/20	1.59/40
Bomb-b50-t50	0.56/50/50	0.47/50/50	–	46.16/100
Bomb-b100-t100	3.13/100/100	5.77/100/100	–	–

Times t in seconds, search space size $|S|$ (number of evaluated states), plan length l. Dashes indicate time-outs.

and with 10 toilets it already performs a lot worse than Conformant-FF. KACMBP remains faster than Conformant-FF, but still its loss of efficiency is significant. In the extreme case displayed in the bottom part of Table 1, there as many toilets as bombs. Under these circumstances, MBP and KACMBP both fail to scale up, while Conformant-FF solves even the task with 100 bombs easily.

Regarding the $h^{\phi(act)}$ version of Conformant-FF, it explores the same number of search states as $h^{\mathcal{I}}$, but it takes considerably more time for doing so. This is due to the time overhead incurred by the more costly SAT tests in the heuristic computation.

Regarding plan length, we observe that Conformant-FF and MBP find the shortest possible plans, making use of additional toilets—if there are $t > 1$ toilets then one can save time by dunking bombs into all of them, and thus sparing some flushing actions when no more bomb will be dunked into a toilet. In difference to this, KACMBP's plans flush the toilet after every dunking action.

The observed runtime behavior is, in fact, relatively easy to explain. As the number of toilets increases, MBP and KACMBP get in trouble due to, presumably, the exponentially increasing number of states in the belief space. For Conformant-FF, however, the more toilets there are the more accurate does its heuristic become. One can observe this in the data when comparing the plan lengths l to the search space sizes $|S|$. In the top part of the table, $|S|$ is a lot larger than l. In the bottom part, however, both numbers are equal. The latter means that, to find the plan, the only states heuristically evaluated by Conformant-FF are those along the search path corresponding to the plan! That is, in Conformant-FF's Hill-climbing procedure, every state has a successor (the first one looked at) with a better heuristic value. Namely, if there are (at least) two non-clogged toilets left then dunking a (potentially armed) bomb into one of them yields a state with a one step shorter relaxed plan—that shorter relaxed plan can use the other non-clogged toilet to dispose of all the remaining bombs. If there is just a single non-clogged toilet, then the nearest successor with a shorter relaxed plan is two steps away. This yields the larger search spaces in the top three parts of Table 1.

Table 2 shows our results in our other traditional conformant benchmark domains, i.e. in Cube, Omelette, Ring, and Safe. From a quick glance, one sees that Conformant-FF is generally competitive with MBP, often outperforming it. KACMBP, however, is clearly superior to both other planners in all the domains. We now consider each of the domains in more detail.

In Cube, one is initially located at any point on a 3-dimensional grid with extension $n \times n \times n$. In each dimension one can "move" up or down. When moving against a border of the grid, nothing happens. We created two different test

Table 2
Results in the conformant benchmarks except Bomb

Instance	Conformant-FF		MBP	KACMBP
	$h^{\mathcal{I}}$	$h^{\phi(act)}$		
	$t/\|S\|/l$	$t/\|S\|/l$	t/l	t/l
Cube-corner-3	0.01/6/6	0.01/7/7	0.02/8	0.00/6
Cube-corner-5	0.03/12/12	0.07/13/13	2.66/16	0.00/12
Cube-corner-7	0.12/18/18	0.46/19/19	–	0.00/18
Cube-corner-9	0.35/24/24	1.31/25/25	–	0.00/24
Cube-corner-11	0.79/30/30	2.76/31/31	–	0.00/30
Cube-center-3	0.06/93/15	0.03/61/9	0.03/9	0.00/11
Cube-center-5	16.98/2211/45	0.82/262/30	2.80/18	0.04/20
Cube-center-7	–	5.81/825/55	–	0.07/31
Cube-center-9	–	80.61/2052/97	–	0.13/40
Cube-center-11	–	1049.74/4913/147	–	0.16/46
Omelette-3	0.03/79/NS	0.01/79/NS	1.38/NS	0.03/NS
Omelette-5	0.09/161/NS	0.04/161/NS	13.46/NS	0.08/NS
Omelette-10	0.73/468/NS	0.25/468/NS	507.12/NS	0.23/NS
Omelette-15	1.59/917/NS	0.87/917/NS	–	0.32/NS
Omelette-20	3.29/1551/NS	2.11/1551/NS	–	0.53/NS
Ring-2	0.02/18/7	0.02/22/12	0.00/7	0.00/5
Ring-3	0.17/36/15	0.21/41/18	0.00/12	0.00/8
Ring-4	2.69/66/26	25.13/127/25	0.00/17	0.00/11
Ring-5	–	–	0.03/22	0.00/14
Ring-6	–	–	0.06/27	0.01/17
Safe-5	0.00/5/5	0.00/5/5	0.01/5	0.00/5
Safe-10	0.05/10/10	0.07/10/10	0.15/10	0.01/10
Safe-30	3.54/30/30	3.79/30/30	–	0.06/30
Safe-50	81.58/50/50	90.68/50/50	–	0.16/50
Safe-70	407.91/70/70	409.93/70/70	–	0.35/70

Times t in seconds, search space size $|S|$ (number of evaluated states), plan length l. Dashes indicate time-outs, NS indicates "no solution", i.e. proved unsolvable.

suites, Cube-corner and Cube-center, where the goal is to move into a corner and the center of the grid, respectively. To get to a corner, it suffices to move n steps in each dimension, in direction of the corner. To get to the center, one must first make sure to be in some corner, then move back into the center from there. The Cube-corner-n/Cube-center-n instances in Table 2 are those with extension n.

In Cube-corner, we observe that both Conformant-FF versions scale well, if not competitively with the extreme efficiency of KACMBP. MBP fails to scale up. Regarding plan lengths, $h^{\mathcal{I}}$-Conformant-FF and KACMBP find the optimal plans. Using $h^{\phi(act)}$, the plans contain one unnecessary action (this is due to a minor implementation detail). MBP's plans are somewhat longer. It is interesting to observe that, for both Conformant-FF versions, the search space is, again, equal to the length of the plan and thus the smallest search space possible. The runtime spent mostly goes into the computation of state transitions, i.e. of known propositions.

In Cube-center, the behavior of the tested planners is very different. Most strikingly, $h^{\mathcal{I}}$-Conformant-FF completely fails to scale up. Using $h^{\phi(act)}$, performance is better but still a lot worse than in Cube-corner. For both Conformant-FF versions, the performance decrease is mainly due to a much larger search space than before. The behavior of MBP and KACMBP does not change that much; the most significant difference is that, now, it is MBP that finds the optimal plans. KACMBP plan's are a little longer than optimal, Conformant-FF's plans are a lot longer than optimal, most of the time.

Obviously, $h^{\phi(act)}$ is a much better heuristic than $h^{\mathcal{I}}$ in Cube-center. Also obviously, the reason for that must be $h^{\phi(act)}$'s more accurate reasoning about past achievements—there is no other difference between the heuristics. What's less obvious is exactly what phenomenon makes it important to reason about past achievements in Cube-center. Here are some intuitions we got from looking at some relaxed plans in small examples. For the initial state of a Cube-center instance, the relaxed plan (which is equal for both heuristics) is relatively short. When ignoring the delete lists, one can

reach the center of an n-cube (for odd n) with $n-1$ steps in each dimension: the old positions are not deleted, and so, to reach the center from any position, it suffices to do $\lfloor n/2 \rfloor$ steps up and $\lfloor n/2 \rfloor$ steps down. That is, in the relaxation the planner does not realize that it has to move to a corner first. This yields local minima that are hard to explore. They are harder to explore for $h^{\mathcal{I}}$ because, as the number of move actions on the path to a believe state increases, $h^{\mathcal{I}}$'s over-approximating reasoning about these action's consequences is likely to conclude that the goal was already achieved by them. Indeed, with $h^{\mathcal{I}}$ Conformant-FF gets down to a state with empty relaxed plan extremely quickly, and then spends ages trying to find a state that actually satisfies the goal.[14]

In the Omelette domain, n eggs must be broken into a bowl without spoiling the bowl. There are two bowls that can both contain up to n eggs. One can "grab" an egg (one can do this infinitely often, i.e. there is no finite store of eggs), one can "break" the egg into one of the bowls, or one can "clean" a bowl, i.e. dispose of its contents. One can also "pour" the contents of one bowl into another, with the obvious effect depending on the fill status of both bowls. Breaking an egg into a bowl spoils the bowl by a non-deterministic effect (namely, when the egg is bad, which one does not know without breaking it). As said, we haven't yet implemented support for such effects so we have modeled that effect as a conditional effect that has a new, unknown, "dummy" proposition as its condition. Note that this is different from a truly non-deterministic effect in that the outcome of the effect will be the same for all eggs, only we don't know what this constant outcome is. Still, there is no conformant plan to our Omelette tasks. We used the "dummy" encoding in the input tasks for all tested planners. The instances Omelette-n in Table 2 are, of course, those with n eggs in the goal.

Once again KACMBP scales—i.e., proves these tasks unsolvable—best, followed closely by Conformant-FF and not so closely by MBP. Note that proving unsolvability involves, for all the systems here, exhausting the space of reachable belief states. One interesting observation is that $h^{\mathcal{I}}$ and $h^{\phi(act)}$ share the same search space—the dead ends detected by both heuristics, i.e. the states with heuristic value ∞, are the same. Unexpectedly, $h^{\phi(act)}$ is a little faster. The heuristic computation of $h^{\mathcal{I}}$ involves traversing longer paths in the *Imp* tree (when finding leafs of this tree), since that tree goes back all the way to the initial state. This takes more time than the state-formula CNF reasoning done by $h^{\phi(act)}$. Probably this time overhead is just due to our relatively naive implementation of traversing paths in the *Imp* tree. We remark that, using h^{inc}, the runtime is roughly similar to that of $h^{\mathcal{I}}$, but a few more states (precisely, $n+2$ more states) are explored since h^{inc} cannot detect dead ends.

In the Ring domain, there are n rooms through which one can "move" in a cyclic fashion. The initial location is unknown. Each room has a window which is either open or closed or locked. The initial states of the windows are unknown. One can apply an action "close" which closes the window of the room in which one is currently located, and one can apply an action "lock"which locks the window of the room in which one is currently located, given the window is already closed. A solution plan moves once through the ring and applies the "close" and "lock" actions after each move. The instances Ring-n in Table 2 are those with n rooms. MBP and KACMBP scale very well, Conformant-FF does not, independently of the heuristic used. KACMBP finds the optimal plans, MBP's plans are a little longer, Conformant-FF's plans are considerably longer.

The poor runtime performance of Conformant-FF here is, partly, due to the oddity in repeated states checking that we mentioned at the end of Section 5. No proposition becomes known until the entire task is solved, every visited state is stored in the same hash table entry, and the number of state comparisons is a square function in the search space size. Turning the repeated states check off, and using only the stagnating states check instead, Conformant-FF with $h^{\mathcal{I}}$ solves the five Ring examples with 0.00/18/7, 0.04/50/20, 0.29/88/39, 1.79/139/64, 9.55/201/95 seconds/search states/actions, respectively. Section 9 outlines some ideas on alternative ways to avoid the overhead of full repeated states checking.

In the Safe domain, a safe has one out of n possible combinations, and one must "try" all combinations in order to assure the safe door is open. The instances in Table 2 are those with n combinations. Once again, KACMBP is extremely efficient. Conformant-FF is a lot slower, MBP can solve only the two smallest examples within the time limit. All planners find the optimal n-action plans trying all combinations in some order. There is no significant difference between the performance of $h^{\mathcal{I}}$ and $h^{\phi(act)}$, which both return perfect (equal to the real goal distance) heuristic values. Like in Ring, most of the runtime goes into the repeated states checking, because the first time a proposition ("open-

[14] A possible way out of this dilemma would be to extend Conformant-FF with mechanisms structuring the search based on "necessary knowledge", as is done in KACMBP [19]: the "necessary knowledge" to reach the center of the cube is the precise position; posing this knowledge as a goal means to move into a corner, so a decomposition of this sort would likely solve Conformant-FF's difficulties in Cube-center.

safe") becomes known is when the task is solved. With only stagnating states checking, $h^{\mathcal{I}}$-Conformant-FF solves the tasks with 0.00/5/5, 0.00/10/10, 0.16/30/30, 1.59/50/50, 6.27/70/70 seconds/search states/actions, respectively.

6.4. Enriched classical benchmarks

Table 3 provides our experimental results in the enriched Blocksworld and Logistics domains. A quick glance at the results reveals that Conformant-FF clearly outscales both MBP and KACMBP in all our test suites.

In all our classical domains enriched with uncertainty, we scale the instances in terms of an *uncertainty parameter*, denoted "U" in the names of the instances, and in terms of various size parameters as appropriate in the domain. For each domain, we have three test suites, with U values 2, 3, and 4, respectively (more details are below).

The Blocksworld domain we use is the variant with three operators to put a block x from another block y onto the table ("move-to-table"), to put a block x from a block y onto a different block z ("move"), and to put a block x from the table onto a block y ("move-from-table"). The uncertainty in our test suites is that the top U blocks on each initial stack (which may be the whole stack) are arranged in an unknown order. Putting a block x from a block y onto the table has conditional effects: (only) if x is located on y in the state of execution, x is put onto the table, and y is cleared. That is, (only) if x is on y then x, including the whole stack of blocks on top of it, is moved to the table. Across our three test suites, the examples are generated with the software by Slaney and Thiebaux [20]. The instances named Blocksworld-UU-bb in Table 3 contain b blocks. Conformant-FF shows fine scalability; the behavior of $h^{\mathcal{I}}$

Table 3
Results in the enriched Blocksworld and Logistics domains

Instance	Conformant-FF		MBP	KACMBP
	$h^{\mathcal{I}}$	$h^{\phi(act)}$		
	$t/\|S\|/l$	$t/\|S\|/l$	t/l	t/l
Blocksworld-U2-b5	0.00/9/7	0.00/9/7	0.75/7	0.35/5
Blocksworld-U2-b6	0.02/19/10	0.01/19/10	16.01/10	0.53/17
Blocksworld-U2-b7	0.03/31/13	0.04/31/13	1148.25/14	1.24/11
Blocksworld-U2-b13	0.44/93/20	0.42/93/20	–	–
Blocksworld-U2-b20	3.75/310/34	3.77/310/34	–	–
Blocksworld-U3-b5	0.00/7/7	0.01/7/7	0.53/8	0.10/8
Blocksworld-U3-b6	0.01/16/12	0.01/16/12	10.62/12	1.08/29
Blocksworld-U3-b7	0.03/30/16	0.04/30/16	–	–
Blocksworld-U3-b13	0.75/136/32	0.70/136/32	–	–
Blocksworld-U3-b20	5.49/423/42	5.37/423/42	–	–
Blocksworld-U4-b5	0.01/7/7	0.00/7/7	0.74/7	0.10/7
Blocksworld-U4-b6	0.08/22/19	0.08/22/19	19.15/19	1.18/11
Blocksworld-U4-b7	0.13/28/23	0.13/28/23	1028.14/21	1.35/13
Blocksworld-U4-b13	2.92/83/43	4.73/83/43	–	–
Blocksworld-U4-b20	1.96/171/48	1.87/171/48	–	–
Logistics-U2-c2-p2-a1	0.00/12/11	0.00/12/11	1.56/13	0.14/17
Logistics-U2-c2-p4-a1	0.01/23/23	0.02/23/23	31.71/25	3.15/49
Logistics-U2-c3-p2-a1	0.01/26/17	0.01/26/17	38.16/17	0.80/27
Logistics-U2-c3-p3-a2	0.03/48/26	0.04/48/26	–	615.95/196
Logistics-U2-c10-p10-a10	4.81/298/81	4.52/298/81	–	–
Logistics-U3-c2-p2-a1	0.03/27/19	0.03/27/19	13.87/22	0.66/23
Logistics-U3-c2-p4-a1	0.05/39/32	0.04/39/32	908.11/54	–
Logistics-U3-c3-p2-a1	0.03/32/20	0.02/32/20	780.19/28	6.41/29
Logistics-U3-c3-p3-a2	0.05/43/26	0.06/43/26	–	–
Logistics-U3-c10-p10-a10	12.90/437/110	14.83/437/110	–	–
Logistics-U4-c2-p2-a1	0.01/21/17	0.01/21/17	76.34/28	1.25/24
Logistics-U4-c2-p4-a1	0.14/57/42	0.16/57/42	–	–
Logistics-U4-c3-p2-a1	0.05/31/18	0.04/31/18	–	60.13/36
Logistics-U4-c3-p3-a2	0.16/77/41	0.18/77/41	–	–
Logistics-U4-c10-p10-a10	26.95/555/114	31.70/555/114	–	–

Times t in seconds, search space size $|S|$ (number of evaluated states), plan length l. Dashes indicate time-outs.

Table 4
Results in the enriched Grid domain, random instances

Instance	Conformant-FF		MBP	KACMBP
	$h^{\mathcal{I}}$	$h^{\phi(act)}$		
	$t/\|S\|/l$	$t/\|S\|/l$	t/l	t/l
Grid-U2-x3-y3-s2-k11-l22	0.03/27/14	0.06/27/14	MEM	1.17/26
Grid-U2-x3-y3-s2-k11-l22	0.05/33/16	0.08/33/16	MEM	10.75/54
Grid-U2-x3-y4-s2-k12-l23	0.11/53/24	0.25/53/24	MEM	783.82/48
Grid-U2-x3-y4-s2-k12-l23	0.26/114/27	0.59/114/27	MEM	–
Grid-U2-x3-y3-s2-k11-l22	0.22/104/28	0.56/104/28	MEM	–
Grid-U3-x3-y3-s3-k111-l111	0.00/13/8	0.01/13/8	MEM	10.36/14
Grid-U3-x3-y4-s3-k112-l112	0.07/44/21	0.22/44/21	MEM	254.12/49
Grid-U3-x3-y4-s3-k112-l112	0.12/48/24	0.33/48/24	MEM	–
Grid-U3-x3-y3-s3-k111-l111	0.13/62/28	0.28/62/28	MEM	–
Grid-U3-x3-y3-s3-k111-l111	0.60/223/40	1.44/223/40	MEM	–
Grid-U4-x3-y4-s4-k1111-l1111	0.02/8/8	0.03/8/8	MEM	88.53/8
Grid-U4-x3-y4-s4-k1111-l1111	0.02/12/12	0.06/12/12	MEM	–
Grid-U4-x3-y4-s4-k1111-l1111	0.04/18/12	0.08/18/12	MEM	–
Grid-U4-x3-y4-s4-k1111-l1111	0.24/60/25	0.70/60/25	MEM	–
Grid-U4-x3-y4-s4-k1111-l1111	0.29/64/30	0.83/64/30	MEM	–

Times t in seconds, search space size $|S|$ (number of evaluated states), plan length l. Dashes indicate time-outs, MEM indicates out of memory.

and $h^{\phi(act)}$ is nearly identical. MBP and KACMBP do not scale up. No planner is clearly superior in terms of plan length.

Our Logistics domain is the following modification of the well-known standard encoding. The uncertainty we introduced lies in that the initial position of each package *within its origin city* is unknown. Loading a package onto a truck has a conditional effect that only occurs when the package is at the same location as the truck. The amount of uncertainty increases with the size of the cities, i.e. city size is our uncertainty parameter U. The instances named Logistics-UU-cc-pp-aa in Table 3 contain c cities (of size U), p packages to be transported, and a airplanes. In each city, there is a single truck. All the examples are randomly generated by uniformly choosing the initial positions of trucks and airplanes, and the initial positions and goal positions of packages. The first four instances in each of the test suites were chosen relatively small in order to be able to obtain some plans with MBP. The largest examples, with 10 cities, packages, and airplanes, were chosen to show Conformant-FF's fine scalability. Again there is not much difference between using $h^{\mathcal{I}}$ or $h^{\phi(act)}$; the latter is a little slower in most examples (particularly, in the largest one). The plans found by MBP or KACMBP are, in most cases, considerably longer than those found by Conformant-FF.

Table 4 shows our comparative results in the enriched Grid domain. The Grid domain was used in the 1st international planning competition (alongside AIPS-98). A robot moves on a 2-dimensional grid on which positions can be locked and, to be accessible, must be opened with a key of a matching shape. The robot can hold one key at a time, and the goal is to transport some keys to specified goal positions. We introduced uncertainty about the locked positions on the grid. The shape of these locks was specified to be an unknown one out of U possible shapes. Opening a lock with a key has a conditional effect that occurs only if the key is of the same shape as the lock. One must thus try all possible keys in order to make sure that a lock is opened. The instances parameters of this domain are more complicated than those of the domains we have seen previously, and so the names of the instances, as show up in Table 4, are a little complicated. Grid-UU-xx-yy-ss means that the grid has x-extension x and y-extension y, and there are s different shapes. The parameters indicated by "-k" and "-l" provide s numbers: the number of keys and locks of each individual shape, namely. So in Grid-U2-x3-y3-s2-k11-l22 there are one key, and two locks, of each of the two shapes. We generated our instances randomly, using software that uniformly distributed the initial positions of robot, keys, and locks, as well as the goal positions of the keys. Of course, such a generation process can result in unsolvable examples—e.g. if, in order to get to a key with a certain shape, the robot has to move across a lock with the same shape. Since the examples we generated for the experiment were relatively small, Conformant-FF could determine unsatisfiability within a few seconds for all the tasks we generated. For the test suites, we used the first few solvable tasks that came up. As one can see in Table 4, the test examples within the suite for each U value do not differ very much in terms of their size parameters. The difficulty of instances in Grid varies a lot already with the distribution of

Table 5
Results in the enriched Grid domain, AIPS'98 instances

Instance	h^{inc} $t/\|S\|/l$	$h^{\mathcal{I}}$ $t/\|S\|/l$	$h^{\phi(act)}$ $t/\|S\|/l$
Grid-U2-AIPS98-1	0.09/25/16	0.16/25/16	0.23/25/16
Grid-U2-AIPS98-2	1.71/198/61	1.71/198/61	4.50/198/61
Grid-U2-AIPS98-3	3.39/386/83	3.74/386/83	13.63/386/83
Grid-U2-AIPS98-4	3.86/368/60	3.83/368/60	7.79/368/60
Grid-U2-AIPS98-5	150.37/1516/224	163.47/1516/224	–
Grid-U3-AIPS98-1	0.28/52/24	0.46/52/24	1.19/52/24
Grid-U3-AIPS98-2	5.14/378/96	5.96/378/96	36.78/472/91
Grid-U3-AIPS98-3	7.43/548/83	8.68/548/83	40.72/548/83
Grid-U3-AIPS98-4	10.44/692/79	11.50/692/79	37.31/692/79
Grid-U3-AIPS98-5	782.55/3744/279	832.01/3744/279	–
Grid-U4-AIPS98-1	0.73/71/34	0.71/71/34	2.43/71/34
Grid-U4-AIPS98-2	25.44/943/127	29.45/943/127	485.62/1870/138
Grid-U4-AIPS98-3	14.86/763/97	18.12/763/97	123.75/763/97
Grid-U4-AIPS98-4	18.70/1102/91	21.92/1102/91	87.79/1102/91
Grid-U4-AIPS98-5	759.44/2477/289	828.39/2477/289	–

Times t in seconds, search space size $|S|$ (number of evaluated states), plan length l. Dashes indicate time-outs.

the locks and keys. We ordered the instances in terms of their difficulty as indicated by the length of the plan found by Conformant-FF. As above in Blocksworld and Logistics, both Conformant-FF versions behave very similar, sharing the same search space; in difference to before, $h^{\phi(act)}$ is quite clearly slower than $h^{\mathcal{I}}$. In all examples, MBP runs out of memory before starting the actual search; the largest instance we could solve with MBP had only 4 positions on the Grid. KACMBP can solve some of the smaller instances, finding comparatively long plans except in one case.

In our random generation process for Grid, the need for filtering unsolvable examples limits the example size severely. To show that Conformant-FF really scales up in the domain, we therefore also ran an experiment with the five Grid instances used in the AIPS-98 competition. These examples are rather large. In each of them there are 4 different shapes. They feature 5×5, 6×6, 7×7, 8×8, and 9×9 grids, respectively; there are 9, 10, 11, 12, and 13 keys; there are 8, 8, 10, 9, and 19 locks. As before, we introduced uncertainty into the instances by giving every lock a range of U possible shapes, U increasing from 2 to 4. We did not try to run MBP and KACMBP on these large examples. Instead we show results for all three variations of Conformant-FF's heuristic function, since they all differ considerably in these challenging tasks. The data are in Table 5.

The main difference between the heuristic functions lies in the speed of computation. In all except two cases—Grid-U3-AIPS98-2 and Grid-U4-AIPS98-2—the search spaces (and returned plans) are the same. Comparing h^{inc} to $h^{\mathcal{I}}$, skipping the SAT reasoning in the heuristic computation yields a small but consistent time advantage. Comparing $h^{\mathcal{I}}$ to $h^{\phi(act)}$, doing the SAT reasoning with the initial state formula, rather than with the formula describing the considered belief state, yields huge savings. Note that the examples become considerably harder to solve as uncertainty increases (with the single exception of Grid-U3-AIPS98-5 and Grid-U4-AIPS98-5). The plan length increases in all cases. With more uncertainty, more keys must be transported to open the necessary locks.

All in all, from our results in the enriched classical domains, we conclude that our approach has the potential to combine the strengths of FF with conformant abilities in domains that combine classical and conformant aspects. In this respect, Conformant-FF is superior to MBP and KACMBP. We consider this an important advantage because it seems likely that, in real-world domains, classical and conformant (uncertain) aspects typically occur together.

6.5. Comparison to GPT and POND

We also ran some experiments with GPT and with POND. We briefly summarize the results, first for GPT then for POND.

GPT is known to be slower than MBP and KACMBP in the traditional conformant domains. We wanted to verify that this is true for the mixed domains, too. Thus, we conducted a number of experiments with GPT on a sample of mixed domains instances. In almost all of them, GPT was slower than MBP and KACMBP, and much slower

than Conformant-FF. In the three smallest Blocksworld-U2 examples, the per-instance runtimes (pure solution times excluding parsing and compilation) we got are 45.77, 112.91, and 1025.33 seconds for GPT as compared with 0.75, 16.01, and 1148.25 for MBP, 0.35, 0.53, and 1.24 for KACMBP, and a few milliseconds for Conformant-FF. For the smallest three instances of Logistics-U2, GPT takes 1.89, 37.19, and 415.44 seconds, MBP takes 1.56, 31.71, and 38.16 seconds, KACMBP takes 0.14, 3.15, and 0.80 seconds, and Conformant-FF takes a few milliseconds. In the Grid suites, GPT kept producing error messages for reasons we could not figure out. These results clearly indicate that, to a varying degree, all of MBP, KACMBP, and Conformant-FF are superior to GPT in these domains.

POND's input language for conformant planning is very similar to that of Conformant-FF; all one needs to do is to replace some keywords in the problem descriptions. We ran POND on all the test examples used in the detailed experiments above, but could get it to find plans in only a fairly small subset of these. If that is due to details in our encoding of the examples, or to inefficiencies in POND's implementation, or to real weaknesses of POND's approach (more on that approach below in Section 8), is hard to tell.

In the Cube examples, POND terminated with a segmentation fault. In most of the other test suites, in the larger examples POND ran out of memory during the pre-process that builds its "labelled uncertainty graph" (*LUG*), the structure that underlies its heuristic computations (see also Section 8). In the small examples that were solved, that pre-process terminated very quickly. The detailed results were as follows (we do not repeat the data for the other planners, given in the tables above).

- In Bomb, the LUG ran out of memory on all examples with more than 10 bombs. The other examples, Bomb-t1-b5, Bomb-t10-b10, Bomb-1-5, Bomb-1-10, Bomb-5-10, Bomb-10-5, were solved in 1.08, 59.01, 0.20, 1.63, 19.10, and 4.23 seconds, respectively. This is considerably slower than either of Conformant-FF, MBP, and KACMBP; compare Table 1. The number of plan steps was always twice the number of bombs, i.e. as with KACMBP no use of additional toilets is made.
- In Omelette, the LUG ran out of memory even in the smallest example.
- The smaller four instances of Ring were solved in 0.13/6, 0.64/11, 14.18/15, and 694.91/15 seconds/steps, respectively. On the largest instance, POND ran out of time. That is, here POND scaled better than Conformant-FF, finding good-quality plans.
- The five Safe instances were solved in 0.13, 0.18, 7.12, 90.77, and 549.54 seconds, with optimal plans. Here, POND's performance is roughly similar to that of Conformant-FF.
- In Blocksworld, the LUG ran out of memory on the two larger examples in each suite, i.e., on those with more than 7 blocks. In the smaller examples, POND is competitive with MBP and KACMBP(solving one more example), but lags far behind Conformant-FF. Precisely, the three smaller Blocksworld-U2 examples are solved in 2.32/6, 16.91/10, and 65.18/12 seconds/steps. In Blocksworld-U3 these numbers are 1.95/7, 10.00/9, and 48.63/15. In Blocksworld-U4 they are 1.67/8, 32.74/19, and 206.13/24.
- In Logistics, the LUG ran out of memory in all but three of the smallest examples. The three solved examples are Logistics-U2-c2-p2-a1, Logistics-U2-c2-p4-a1, and Logistics-U3-c2-p2-a1. They are solved in 3.73/7, 16.00/11, and 41.55/10 seconds/steps. Compared to the other planners here, POND is relatively slow but finds very good plans (the best one found, in all three cases).
- In Grid, the LUG ran out of memory even in the smallest example.

7. Non-deterministic effects

If an effect is marked as non-deterministic, and is applied to a world state w, then the outcome are two possible successor world states w' and w''—one where the effect was applied, one where it was not applied. In the presence of such effects, a conformant plan must achieve the goals for all possible initial world states *and* for all possible outcomes of the non-deterministic effects. Formulated in terms of belief states, the criterion that a conformant plan has to meet is still that it ends in a belief state s where all world states w in s contain the goals. We have not yet implemented support for non-deterministic effects in Conformant-FF. While the needed extensions are very easy for the computation of known propositions, and for the computation of relaxed plans, it is unclear how to adequately extend our repeated states checking. Doing the latter is a topic for future work. Here, we outline the main issues arising in that important context.

The computation of known propositions can be extended to non-deterministic effects as follows. When generating the formula $\phi(act)$ encoding the semantics of an action sequence *act*, simply skip non-deterministic effects when inserting the effect axioms, i.e., insert those for deterministic effects only. The rest of the formula's definition, as well as the SAT calls made to check for known/negatively known propositions, can remain exactly the same. To see this, reconsider Proposition 2. With non-deterministic effects, there are now several satisfying assignments to $\phi_I(act)$—namely, corresponding exactly to all possible outcomes of the non-deterministic effects. (When non-deterministic effects do not enforce their adds and deletes by effect axioms, but the frame axioms for the respective proposition values respect these effects, then consequently the affected variables can take on both truth values in the next time step.) Still, the formula *implies* a proposition value at a time step only if the value holds at this step in *all* possible outcomes of the action sequence, which is exactly what we need to know about.

To extend the relaxed planning process to non-deterministic effects, it suffices to ignore these. A non-deterministic effect cannot be used to establish a proposition, and in the relaxed setting a non-deterministic effect, like the deterministic ones, cannot interact with the propagation of a positive truth value. Note that we make no claims whatsoever about how this technique, and the technique described above, will behave in practice; this has to be tested empirically.

In the repeated states checking, one can, in theory, proceed with an extension just as simple as the ones outlined above. Consider the belief state equivalence test. Say we simply leave the entire machinery unchanged, using the extended definition of the formulas $\phi(act)$. Say $\phi_{I=}(act) \wedge \phi_{I=}(act') \wedge \neg p(act) \wedge p(act')$ is unsatisfiable. With the extended definition of the formulas $\phi(act)$, this means that for all possible initial world states, for all possible outcomes of the non-deterministic effects, p can't be false after *act* but true after act'. So by doing exactly the same SAT calls as before, one gets a sufficient test for belief state equality. The test succeeds iff, starting from every possible initial world state, the resulting world states in s and s' are the same *independently of the outcomes of the non-deterministic effects*. Note that this is a very strong requirement. It is the nature of non-deterministic effects to result in several different world states. Unless these different world states are merged back into a single one by later actions in the sequence, the test will fail. Consider the following example. The initial state formula is empty, and there is a single action a that has no other effects than non-deterministically adding a proposition p. Obviously, the belief state after $act = \langle a \rangle$ is equal to the belief state after $act' = \langle a, a \rangle$. The reached world states are $\{p\}$ and $\emptyset$ in both cases. However, the formula $\phi_{I=}(act) \wedge \phi_{I=}(act') \wedge \neg p(act) \wedge p(act')$ asks if it can be the case that p is false after *act* but true after act'. And of course that can happen—namely, when a's effect does not occur on *act*, but does occur on act'. Note that the SAT test would fail (the formula would be satisfiable) even if we were to compare *act* to itself.

Similar difficulties as above can be observed with the (straightforward extension of the) belief state domination test. The problem with the SAT-based equivalence/domination tests is that, in the presence of non-deterministic effects, the formulas ϕ encode a *set* of world states (rather than just a single world state) for every possible initial world state I. This makes the SAT tests lose precision. As said, developing repeated states checking methods that are practically useful, in the presence of non-deterministic effects, is a topic for future work.

8. Related work

A better understanding of Conformant-FF can be obtained by examining its relation to a number of strands of work on conformant planning. Below we describe the three main strands of research on conformant planning, and then place Conformant-FF in their context.

CGP [21] is the first specialized conformant planner. CGP takes the essential ideas behind the Graphplan [15] algorithm and applies them in the context of conformant planning. For its forward reachability phase, CGP builds one planning graph for each possible initial world state, while recording mutexes within each graph and between graphs. It searches backward for a solution on these planning graphs. Action choices made in one graph are propagated to the others during this process. However, it has difficulty scaling up when the number of initial states is large. Thus, we can characterize CGP's approach as based on utilizing representation and solution methods from classical planning for reasoning about individual initial states, while propagating information between these solutions in a sophisticated way. A close relative of this is Frag-Plan [22], which generates plans (using planning-graphs) for concrete initial states, attempting to first tackle initial states for which finding a plan is harder. The initial plan is then extended to cover all possible initial states by including fragments from plans for different initial states. This requires some back-and-forth reasoning between different states: while adapting an existing plan to work for a new initial state s we may make it inappropriate for some previously handled initial state s'. While fixing this problem, we may introduce a

problem somewhere else, etc. C-Plan [8] is related to CGP and Frag-Plan in the sense that it also relies on classical planning techniques to a large extent. Namely, it uses sat-based (classical) planning techniques to generate candidate plans: a candidate plan is a satisfying assignment to the initial state formula plus the usual SAT encoding, meaning that the candidate is a plan for at least one possible initial world (the one chosen by the assignment). It is then tested (by another SAT test) whether the candidate is, in fact, a conformant plan. Like CGP and Frag-Plan, this approach is likely to have trouble scaling in the number of initial states—one would expect the number of candidate plans to be closely related to the number of initial states, although the relationship can be complicated, in general. Unlike CGP and Frag-Plan, C-Plans generate-and-test approach has no backtracking element; the candidate plans are viewed separately and no interactions between them are explored.

The most successful strand of work on conformant planning is based on the idea of search in the space of belief states. Planners from this breed differ in two central parameters: how they represent belief-states and how they search in this space. In particular, among these planners, the most successful ones employ some form of heuristic search. The first planner to utilize belief-space search is GPT [1]. GPT represents belief-states explicitly, and searches this space using A*-based forward heuristic search. The heuristic utilizes relaxed dynamic programming techniques to compute reachability and distance estimates. GPT finds optimal plans, but has problems scaling up because, first, its heuristic is often not informative enough and, second, aside from the size of the belief *space*, the size of the belief *states* can be prohibitive even in relatively simple conformant planning problems. To remedy this problem, CMBP [2] utilizes symbolic, BDD-based data structures for representing belief states, and uses BDD manipulation techniques to perform state update. This change leads to a much more compact and manageable approach for planning in belief space. However, CMBP performs uninformed breadth-first search, and so its ability is still limited, like GPT's, through the size of the belief space. This problem was addressed by HSCP [23]. HSCP uses symbolic machinery much like CMBP to perform backward heuristic search. That is, it performs regression at the level of belief state representation. The regression of a belief-state b over an action a is the largest belief-state for which by applying a we get to a belief state that is contained in b. The heuristics used there is based solely on the cardinality of the belief state. While better than CMBP, this heuristic is somewhat simplistic. A more advanced planner related to HSCP is KACMBP [19]. KACMBP uses forward search. Thus, it is informed by the initial state and considers reachable states only. KACMBP also uses a more sophisticated heuristic approach. Although it also considers the cardinality of belief states, it does not do so blindly. KACMBP attempts to actively achieve knowledge that it deems useful by preferring actions that lead to states with more desirable knowledge.[15]

CAltAlt and POND [7,24] are two more recent planners that utilize heuristic search over belief states. CAltAlt searches backwards, whereas POND searches forward. Like CMBP and its descendants, POND uses BDDs to represent the current belief state, while CAltAlt uses a clause-based representation. However, they are also closely related to CGP in that they make extensive use of planning graphs in generating their heuristics estimates. Indeed, the main novel ideas in these planners are their more informed heuristic estimates and the data-structures used to obtain them. In particular, they introduce a labeled uncertainty graph, a planning graph-like structure that stores in one planning graph information that CGP obtains from multiple planning graphs. This is done by labeling certain facts with the initial state conditions that support them (represented using BDDs), and propagating these labels over the actions. However, unlike CGP, the planning graph is not used to actually perform the search.

Finally, there is a third strand of work on conformant planning which we shall refer to as the logic-based approach. What is common to the set of planners in this (more loosely related) category is the reduction of conformant planning to inference tasks in existing logical formalisms. We have already discussed the C-Plan planner which uses SAT techniques. C-Plan generates candidate plans using planning-as-satisfiability based methods, and then uses unsatisfiability checks to see whether they are actually conformant plans. QBFPLAN [25] uses the more powerful (and more complex) formalism of quantified boolean formulas. That is, it reduces the question of the existence of a (bounded-length) conformant plan to the satisfiability of a quantified boolean formula. A different reduction is used by DLV^k [26]. It reduces (bounded) conformant planning to answer-set programming and uses the rich domain description language $\mathcal{K}$. Somewhat related to this approach is the Petrick and Bacchus' PKSPlan [18] which shares with DLV^k the idea of us-

[15] Note: CMBP was integrated into the more general MBP planner, which supports, in addition, different search methods (forward, backward, heuristic) as well as contingent planning and more. Thus, in the experimental section and throughout much of the paper we speak of MBP, not CMBP. In terms of historical progression of ideas, the above is a more useful description. However, restricted to conformant planning, KACMBP is a more advanced planner than the current MBP.

ing a rich domain description language, albeit a different one. PKSPlan uses logical formulas to represent the current state of *knowledge* of the agent. Its actions explicitly manipulate these knowledge formulas.

In terms of performance, belief-space search planners are the best studied and, probably, the best performers. A good experimental analysis of the more successful approaches appear in [19]. Of these planners, aside from Conformant-FF, KACMBP appears to be the most efficient, outperforming CGP and its related planners. We note that w.r.t. Conformant-FF, their results basically agree with our results: KACMBP is generally better on the traditional conformant domains, whereas Conformant-FF is better on the mixed domains. It is somewhat difficult to compare them to the logic-based planners because of differences in the input languages. For instance, the input language used by DLV^k is much richer than the PDDL-like language Conformant-FF uses, whereas the language used by PKSPlan is somewhat incomparable. The comparisons of CAltAlt and KACMBP in [24] shows that CAltAlt is competitive with KACMBP, in fact, fairing better than it in some rover and logistics problems. However, that comparison does not include many standard test domains (e.g., cube, ring, omelette, safe).

Conformant-FF is best viewed as a belief-space search planner, as it represents the current state of the search process using a structure that is equivalent to a belief state. Its main innovation is the implicit manner in which it describes these belief-states. This description is much more compact than the explicit description used by any of the other belief-space search planners. The price Conformant-FF pays for this space reduction is in the time required to answer queries about each belief state. However, as evident from our results, this tradeoff often pays off. Another interesting innovation of Conformant-FF is the use of a single planning-graph for computing a relaxed plan. Unlike the multiple planning graphs used by CGP, this single planning graph is able to capture information about multiple initial states, by explicitly differentiating between known and unknown propositions. This allows Conformant-FF to produce a relatively informed heuristic estimate (in fact, a relaxed plan) quickly. CAltAlt and POND also utilize more complex reasoning on top of a single planning graph, but that appears more complex and more costly to construct. Conformant-FF does not take into account issues such as the cardinality of a belief-state or desirable knowledge in this computation. Possibly, by taking these issues into consideration, following KACMBP, a more robust heuristic estimate can be obtained (see also next section). Finally, Conformant-FF is related to the logic-based strand in its use of logical formulas to represent the current belief-state, and the use of (un)sat queries to check properties of this state, which is virtually identical with the use of such checks by C-Plan to determine whether a candidate plan solves the task. Conformant-FF simply makes much more extensive use of these checks, using them to check properties of partial plans, as well.

9. Conclusion

We introduced a new, lazy, approach for representing belief states in conformant planning. Based on that, we extended the classical heuristic forward search planner FF into conformant planning. The necessary knowledge about belief states—the known propositions—is computed via a CNF reasoning technique. The relaxed planning method that FF uses to compute its heuristic function is modified accordingly, with a complete but unsound form of the CNF reasoning about known propositions. The resulting planner Conformant-FF is competitive with the state-of-the-art conformant planners MBP, KACMBP, GPT, and POND, sometimes outperforming all of them. Conformant-FF shows the potential to combine the strength of FF with conformant abilities.

An interesting idea how to overcome Conformant-FF's current limitations is to try and combine the heuristic principles used in Conformant-FF and KACMBP. While the former system tries to minimize estimated goal distances, the latter system tries to minimize belief state size (more precisely, tries to accomplish useful information about the state). The comparative data obtained in our experiments reveals that the two kinds of approaches have very different strengths and weaknesses. KACMBP outperforms Conformant-FF in cases where planning comes close to reducing the initial uncertainty. Conformant-FF becomes superior when more classical planning aspects come into the play. A promising idea is to use a hybrid heuristic (h, b) where h estimates goal distance, b estimates belief state size, and $(h(s), b(s)) < (h(s'), b(s'))$ if $h(s) < h(s')$ or $h(s) = h(s')$ and $b(s) < b(s')$. To improve Conformant-FF, the main question to answer here is if, and how, belief state size can be estimated based on our data structures. Similarly, one can try to improve KACMBP by estimating goal distances based on KACMBP's data structures.

It may be possible to overcome the huge overhead induced by Conformant-FF's repeated states checking, as observed in the Ring and Safe domains, when there is a lack of different known/unknown propositions in the search space. One can introduce incomplete pre-tests to avoid full CNF tests for state pairs s and s' that can quickly shown

to be non-equal/non-dominating. For example, say ϕ is the CNF whose satisfiability must (repeatedly) be tested for deciding whether s is equal to s'. If even a 2-projection of ϕ is solvable, then ϕ is solvable, and s is not equal to s'.

In the extension to non-deterministic effects, as said it is most important (and difficult) to come up with an adequate extension of repeated states checking. One idea that may turn out feasible is to enumerate the outcomes of the non-deterministic effects on the action sequences *act* and *act*$'$. Say it is the case that, for every initial world state, for every outcome of the effects on *act*, there is an outcome of the effects on *act*$'$ so that the resulting world states are equal. Then the belief state represented by *act*$'$ is a superset of the one represented by *act*, and *act*$'$ can be pruned if *act* was visited before. Note that, formulated in terms of a satisfiability test, the outlined test involves a quantifier alternation in the prefix of a CNF formula. Such a test might be best implemented using a QBF solver.

An important observation is that our approach is not inherently restricted, through relying on its "known propositions" computation, to conjunctive goals and action preconditions. If the goal is a DNF formula G, then an action sequence *act* ends in a goal state iff $\phi(act)$ implies G at its end point. That entailment check can be done by conjoining $\phi(act)$ with the negation of G, yielding a new CNF, and seeing whether that new CNF is satisfiable. Similarly, DNF action preconditions can be dealt with. DNF effect conditions can be equivalently split up into a separate effect for each disjunct. Together with the standard pre-processes known from classical planning, compiling quantifiers and negations away [27], with this, one can handle the full scale of ADL syntax (in fact, quantifiers, negations, and disjunctive effect conditions are handled this way already by our current code). Of course, such a compilation process (in particular bringing formulas into DNF) is worst-case exponential and may be prohibitively costly. In classical planning, with some simple optimizations (detecting static predicates, e.g.) the compilation is usually manageable in practice. Whether the same holds true in conformant planning has to be empirically verified.

In a work effort we have already (largely) completed, we extended Conformant-FF to contingent planning, i.e. to handle observations, and to produce plans with branches depending on the observed properties [28]. In the long term, we plan to extend Conformant-FF into a probabilistic planner.

Acknowledgements

We would like to thank Piergiorgio Bertoli and Alessandro Cimatti for their help in using MBP and KACMBP, Blai Bonet for his help with GPT, Daniel Bryce for his help with POND, and Dave Smith for his help in understanding and describing related work. We also thank the anonymous reviewers for their comments, which helped to improve the paper. Ronen Brafman is partially supported by the NASA Intelligent Systems Program, and the Paul Ivanier Center for Robotics and Production Management at Ben-Gurion University. Jörg Hoffmann was, at the time of doing the described work, supported by the DFG (Deutsche Forschungsgemeinschaft), project HEU-PLAN II.

Appendix A. Proofs

This appendix provides the full proofs to (as well as some discussions related to) the theorems contained in Section 4.

We first show that deciding conformant plan existence with empty delete lists is co-NP-complete, given one assumes the generous action execution semantics, i.e., drops the requirement that the actions in the plan have to be applicable at their point of execution no matter what initial world one starts from. (Remember that in the generous semantics, a non-applicable action does not change the world state, while in the non-generous semantics a non-applicable action leads into an undefined world state.)

Theorem 1 *(A^+-complexity). Given a conformant task $(A, \mathcal{I}, G)$ where the delete lists of all effects are empty. Deciding whether there exists a generous plan for $(A, \mathcal{I}, G)$ is co-NP-complete.*

Proof. Hardness follows directly from Proposition 1. We show membership by constructing a propositional CNF formula ϕ (of polynomial size) so that ϕ is unsatisfiable iff there exists a generous plan for $(A, \mathcal{I}, G)$. The idea behind the construction is to encode the semantics of executing all actions in parallel m times, where m is the number of distinct conditional effects in the task. We first explain the construction of the formula ϕ, then we argue why no more than m parallel action steps are needed to achieve the goal, if that is possible.

The formula ϕ is similar to the formulas $\phi(act)$ introduced in Section 3 to encode the semantics of action sequences. We use a time index to differentiate between values of propositions at different points along the hypothetical execution of the parallel action steps. First, as before ϕ contains $\mathcal{I}$ indexed with time 0 (i.e., for each clause $l_1 \vee \cdots \vee l_k \in \mathcal{I}$ we add $l_1(0) \vee \cdots \vee l_k(0)$ into ϕ). Then, for all $0 \leqslant t < m$, for all actions a with precondition $pre(a) = \{p_1, \ldots, p_k\}$, for all effects $e \in E(a)$ with condition $con(e) = \{c_1, \ldots, c_l\}$, and for each $p \in add(e)$, we insert the clause $\neg p_1(t) \vee \cdots \vee \neg p_k(t) \vee \neg c_1(t) \vee \cdots \vee \neg c_l(t) \vee p(t+1)$, i.e. the implication $p_1(t) \wedge \cdots \wedge p_k(t) \wedge c_1(t) \wedge \cdots \wedge c_l(t) \rightarrow p(t+1)$. Also, for all $0 \leqslant t < m$, and for all propositions p, we insert the clause $\neg p(t) \vee p(t+1)$, i.e. the implication $p(t) \rightarrow p(t+1)$. In the absence of delete effects, these clauses obviously capture the semantics of applying all actions m times in parallel (with a generous execution model), except that one can make a proposition $p(t+1)$ true "without a reason" because we have not included any clauses that force p to stay false if it was false at t and was not added by any action effect. But it is also possible to let p stay false in this latter case. We now, finally, add the clause $\neg g_1(m) \vee \cdots \vee \neg g_n(m)$ into ϕ where $G = \{g_1, \ldots, g_n\}$. The overall formula is then unsatisfiable iff, from every possible initial world state, applying all actions in parallel m times forces the goals to be true.

If we fix any possible initial world state I, then no plan needs more than m steps to achieve G. This is simply because, without delete lists, if an action application does not result in the application of any conditional effects that have not already been applied earlier in the action sequence, then the action can be skipped because it does not add any new propositions to the world state it is executed in. Now, say the action sequence act is a generous plan for $(A, \mathcal{I}, G)$. Then, for any possible initial world state $I \in 2^{\mathcal{I}}$, act contains a subsequence act' so that executing act' in I achieves G. With the above, we can assume that the length of act' is at most m. This means that act' is contained in the m parallel action steps encoded in ϕ, so if one sets the $t = 0$ variables in ϕ to the values given by I then one obtains a contradiction to the goals-false clause at m. It follows that, if there is a generous plan for $(A, \mathcal{I}, G)$, then ϕ is unsatisfiable. The other way around, if ϕ is unsatisfiable then obviously a generous plan is to apply all actions in sequence, m times iteratively. This concludes the proof. □

The construction in the proof does not work if one assumes a non-generous execution semantics of actions, because in this case one cannot easily model a "parallel" execution of all actions. The formula ϕ in the proof assumes that, at plan execution time, non-applicable actions in the parallel execution do not harm the plan. More formally, unsatisfiability of ϕ does not imply that there is a non-generous plan.

We now prove that the CRPG algorithm from Fig. 2 works as required. We use the following notation. Given an action sequence $act = \langle a_{-n}, \ldots, a_{-1} \rangle$, and a CRPG built for act up to proposition layer m, for any t with $-n \leqslant t < m$ we denote by *act-CRPG*(t) the action sequence that consists of all action layers $A(-n), \ldots, A(t)$ in an arbitrary linearization; for $t = -n - 1$, the action sequence is assumed empty.

We first prove a lemma containing the main technical argument. Before reading the proof, recall Section 4.2 and Fig. 2. In particular, remember that the action layers contain the actions in their relaxed form (by a relaxation function $|_1^+$), i.e. without delete lists and with all but one of the effect conditions removed. The prove is not overly deep, but a little complicated due to the many details that need to be taken into account.

Lemma 1. *Given a conformant task $(A, \mathcal{I}, G)$, an executable action sequence act, and a relaxation function $|_1^+$ for A. For all proposition layers t built by* build-CRPG$(act, A, \mathcal{I}, G, |_1^+)$*, and for all propositions p, p is known after act-CRPG$(t-1)$ iff $p \in P(t)$.*

Proof. Because act is executable, $act|_1^+$ also has that property, i.e., when executing $act|_1^+$ in any initial world state $I \in 2^{\mathcal{I}}$, the preconditions of the actions are true at the respective points of execution. We prove the following properties, for all propositions p and for all $-n \leqslant t < m$, by induction over t.

(A) For any initial world state $I \in 2^{\mathcal{I}}$, p is true after executing *act-CRPG*$(t-1)$ in I iff either
  (1) there exist $a \in A(t-1)$ and $e \in E(a)$ s.t. $p \in add(e)$ and $con(e) \in P(t-1)$, or
  (2) there exists $l \in Impleafs(p(t))$ s.t. $l \in I$.

(B) p is known after *act-CRPG*$(t-1)$ iff $p \in P(t)$.

(C) p is unknown after *act-CRPG*$(t-1)$ iff $p \in uP(t)$.

Say the length of *act* is n. The base case, $t = -n$, is trivially true for all three properties. Assume the properties hold true for some proposition layer $t - 1 \geqslant -n$. Assume build-CRPG$(act, A, \mathcal{I}, G, |_1^+)$ builds the next proposition layer, time step t. We show that then the properties hold for that layer, too. We consider the three properties from top to bottom.

Property (A)*, from left to right*. Say p is true after *act-CRPG*$(t-1)$. Then p is added by an occurring effect e of an action a at layer $t-1$. That is, we have $a \in A(t-1)$ and $e \in E(a)$ s.t. $p \in add(e)$, and $con(e)$ is true after *act-CRPG*$(t-2)$. If there is one such pair a and e s.t. $con(e) \in P(t-1)$, then property (A)(1) is satisfied and we are done. Otherwise, let a and e be one pair as above. We can conclude two things about a and e:

- First, consider the induction hypothesis on property (A) for $con(e)$ and *act-CRPG*$(t-2)$. Because $con(e) \notin P(t-1)$, property (A)(1) does not hold. Thus property (A)(2) must hold, i.e. there exists $l \in Impleafs(con(e)(t-1))$ s.t. $l \in I$.
- Second, consider the induction hypothesis on property (B) for $con(e)$ and *act-CRPG*$(t-2)$. Because $con(e) \notin P(t-1)$, $con(e)$ is not known after *act-CRPG*$(t-2)$. But $con(e)$ is true after *act-CRPG*$(t-2)$, so we have that $con(e)$ is unknown after *act-CRPG*$(t-2)$. We can now apply the induction hypothesis on property (C) for $con(e)$ and *act-CRPG*$(t-2)$, and we get that $con(e) \in uP(t-1)$. Therefore, by construction of the build-CRPG algorithm, there is an edge $(con(e)(t-1), p(t)) \in Imp$.

With $l \in Impleafs(con(e)(t-1))$ and $(con(e)(t-1), p(t)) \in Imp$ we have $l \in Impleafs(p(t))$, so property (A)(2) holds for p and *act-CRPG*$(t-1)$, and we are done.

Property (A)*, from right to left*. We first prove that property (A)(1) implies truth of p after *act-CRPG*$(t-1)$, then we do the same for property (A)(2).

- Case (1). Say there exist $a \in A(t-1)$ and $e \in E(a)$ s.t. $p \in add(e)$ and $con(e) \in P(t-1)$. First, observe that the preconditions $pre(a)$ of a are true after *act-CRPG*$(t-2)$. If $t - 1 \geqslant 0$, this follows from induction hypothesis on property (B) for (all) $pre(a)$ and *act-CRPG*$(t-2)$. If $t - 1 < 0$, i.e. if a is one of the actions in the sequence *act* leading to our search state, then the build-CRPG algorithm imposed no restriction on a's preconditions. But *act* is executable by prerequisite, and with the argument given at the top of this proof we have that $pre(a)$ is true at the point of a's execution, i.e., after *act-CRPG*$(t-2)$. So a is applicable. By induction hypothesis on property (B) for $con(e)$ and *act-CRPG*$(t-2)$, we have that $con(e)$ is true after *act-CRPG*$(t-2)$. So p is true after *act-CRPG*$(t-1)$, and we are done.
- Case (2). Say there exists $l \in Impleafs(p(t))$ s.t. $l \in I$. Going one step backwards over the edges in *Imp*, we then get that there exists a p' with $(p'(t-1), p(t)) \in Imp$ and $l \in Impleafs(p'(t-1))$. By induction hypothesis on property (A) for p' and *act-CRPG*$(t-2)$, p' is true after *act-CRPG*$(t-2)$. Now, let a and e be the action and effect responsible for the edge $(p'(t-1), p(t)) \in Imp$, i.e. $a \in A(t-1)$ and $e \in E(a)$ with $p \in add(e)$ and $con(e) = p'$. With the same arguments as above in case (1), we have that $pre(a)$ is true after *act-CRPG*$(t-2)$. Since we have already shown that $con(e) = p$ is also true after *act-CRPG*$(t-2)$, it follows that p is true after *act-CRPG*$(t-1)$, and we are done.

Property (B). We prove this by equivalently transforming the left hand side of the property to the right hand side of the property. We first give the sequence of transformation steps, then we explain why these steps are valid.

(a) p is known after *act-CRPG*$(t-1)$ $\Leftrightarrow$
(b) For all $I \in 2^{\mathcal{I}}$, p is true after executing *act-CRPG*$(t-1)$ in I $\Leftrightarrow$
(c) For all $I \in 2^{\mathcal{I}}$, either property (A)(1) holds for p or property (A)(2) holds for p $\Leftrightarrow$
(d) Either property (A)(1) holds for p, or for all $I \in 2^{\mathcal{I}}$ property (A)(2) holds for p $\Leftrightarrow$
(e) Either property (A)(1) holds for p, or $\mathcal{I} \rightarrow \bigvee_{l \in Impleafs(p(t))} l$ $\Leftrightarrow$
(f) $p \in P(t)$

The step from (a) to (b) just inserts the definition of known propositions. The step from (b) to (c) is valid because we have already proved property (A) for p and *act-CRPG*$(t-1)$. Note that, at this point, we need the induction hypothesis as used to prove property (A). The step from (c) to (d) is valid because property (A)(1) does not depend

on I. The step from (d) to (e) is valid because there is a tree leaf l in *every* initial world state iff the initial formula implies the disjunction of the leafs. The step from (e) to (f) is valid by construction of the build-CRPG algorithm: (e) is the precise condition under which the algorithm inserts p into $P(t)$.

Property (C)*, from left to right*. Because p is unknown after *act-CRPG*$(t-1)$, there exists $I \in 2^{\mathcal{I}}$ s.t. p is true after *act-CRPG*$(t-1)$. We already proved property (A) for p and *act-CRPG*$(t-1)$, so we get that either property (A)(1) or property (A)(2) holds for p and I. We show below that

$(*)$ assuming property (A)(1) leads to a contradiction.

So property (A)(2) must hold. That is, there exists $l \in \mathit{Impleafs}(p(t))$ s.t. $l \in I$. This implies, in particular, that there exists an edge $(p'(t-1), p(t)) \in \mathit{Imp}$. By construction of the build-CRPG algorithm, this implies that either $p \in uP(t)$, or $p \in P(t)$. (The latter can be the case if p was inferred due to $\mathcal{I} \rightarrow \bigvee_{l \in \mathit{Impleafs}(p(t))} l$.) But by the already proved property (B) for p and *act-CRPG*$(t-1)$, we have that $p \notin P(t)$, since it would otherwise be known after *act-CRPG*$(t-1)$. It follows that $p \in uP(t-1)$, and we are done.

It remains to show $(*)$. We have that p is unknown after *act-CRPG*$(t-1)$, and we assume that property (A)(1) holds for p and *act-CRPG*$(t-1)$. That is, there exist $a \in A(t-1)$ and $e \in E(a)$ s.t. $p \in \mathit{add}(e)$ and $\mathit{con}(e) \in P(t-1)$. With exactly the same arguments as given above in the right-to-left proof of Property (A), case (1), we can show that a will be applicable, at its point of execution in *act-CRPG*$(t-1)$, from *every* initial world state. Further, with induction hypothesis on property (B) for $\mathit{con}(e)$ and *act-CRPG*$(t-2)$, we have that $\mathit{con}(e)$ is known after *act-CRPG*$(t-2)$. It follows that p is known after *act-CRPG*$(t-1)$, in contradiction. This concludes the argument.

Property (C)*, from right to left*. If $p \in uP(t)$ then, by construction of the build-CRPG algorithm, it follows that there exists a proposition $p' \in uP(t-1)$ with an edge $(p'(t-1), p(t)) \in \mathit{Imp}$. Let a and e be the action and effect responsible for the edge $(p'(t-1), p(t))$, i.e., $a \in A(t-1)$, $e \in E(a)$, $p \in \mathit{add}(e)$, $\mathit{con}(e) = p'$. Observe that a will be applicable, at its point of execution in *act-CRPG*$(t-1)$, when starting from any initial world state: again, this follows by exactly the same arguments as given above in the right-to-left proof of Property (A), case (1). Now, observe that, by induction hypothesis on property (C) for p' and *act-CRPG*$(t-2)$, p' is unknown after *act-CRPG*$(t-2)$. This means in particular that there exists $I \in 2^{\mathcal{I}}$ s.t. p' is true after executing *act-CRPG*$(t-2)$ in I. With applicability of a, it follows that p is true after executing *act-CRPG*$(t-1)$ in I. Thus p is either known or unknown after *act-CRPG*$(t-1)$. Assume p is known after *act-CRPG*$(t-1)$. Then, by the already proved property (B) for p and *act-CRPG*$(t-1)$, we get $p \in P(t)$. But, by construction of the build-CRPG algorithm, $P(t) \cap uP(t) = \emptyset$, in contradiction to our prerequisite $p \in uP(t)$. It follows that p is unknown after *act-CRPG*$(t-1)$. □

Next, we consider the termination criterion in Fig. 2, i.e. the criterion that makes the CRPG algorithm return FALSE. The criterion asks if, in a time step t just built, the sets of known and unknown propositions, as well as the reachable leafs of the *Imp* tree for the latter propositions, have stayed the same from layer t to layer $t+1$. It is relatively easy to see that, in this case, the same would hold true at all future time steps $t' > t$: if there was progress in the future then there would also be some progress from t to $t+1$ already.

Lemma 2. *Given a conformant task* $(A, \mathcal{I}, G)$*, an action sequence act, and a relaxation function* $|_1^+$ *for A. If, for proposition layers t and* $t+1$ *built by* build-CRPG$(\mathit{act}, A, \mathcal{I}, G, |_1^+)$*, it holds that* $P(t+1) = P(t)$*,* $uP(t+1) = uP(t)$*, and* $\forall p \in uP(t+1)$: $\mathit{Impleafs}(p(t+1)) = \mathit{Impleafs}(p(t))$*, then the same conditions would hold for layers* $t+1$ *and* $t+2$*.*

Proof. Assume build-CRPG$(\mathit{act}, A, \mathcal{I}, G, |_1^+)$ would not return FALSE in iteration t, and proceed to iteration $t+1$. With $P(t+1) = P(t)$, $A(t+1) = A(t)$ would hold. With that, with $P(t+1) = P(t)$, and with $uP(t+1) = uP(t)$, the first **for**-loop in the call to build-timestep$(t+1, A(t+1))$ would do exactly the same things as in the previous iteration. Consider the second **for**-loop in the call to build-timestep$(t+1, A(t+1))$. It proceeds over the same propositions p as in iteration t. We now show that, for all these p, $\mathit{Impleafs}(p(t+2)) = \mathit{Impleafs}(p(t+1))$, which then concludes the argument.

We have $\mathit{Impleafs}(p(t+2)) \supseteq \mathit{Impleafs}(p(t+1))$ due to the NOOP action. To see the opposite direction, say we have a proposition l with $l \in \mathit{Impleafs}(p(t+2))$. Then the path of *Imp* edges from l to $p(t+2)$ goes through an edge

$(p'(t+1), p(t+2)) \in Imp$. Obviously, we have $l \in Impleafs(p'(t+1))$. Now, we can show that the same edge and relation to l exist one step earlier in the CRPG.

(1) Observe that $p' \in uP(t+1)$ – if p' was in $P(t+1)$, then p would be in $P(t+2)$.
(2) Because $A(t+1) = A(t)$, the action $a \in A(t+1)$ responsible for the edge $(p'(t+1), p(t+2))$ also occurs in $A(t)$, yielding the edge $(p'(t), p(t+1)) \in Imp$.
(3) With $p' \in uP(t+1)$, we have $Impleafs(p'(t+1)) = Impleafs(p'(t))$. This yields $l \in Impleafs(p'(t))$.

With $l \in Impleafs(p'(t))$ and $(p'(t), p(t+1)) \in Imp$, we get $l \in Impleafs(p(t+1))$, and are done. □

As a side remark, note that we don't need to postulate executability of *act* for Lemma 2 to hold. Lemmas 1 and 2 together allow the conclusion that the CRPG algorithm is complete, i.e., returns FALSE only if there is no relaxed plan.

Theorem 2 *(CRPG-completeness). Given a conformant task $(A, \mathcal{I}, G)$, an executable action sequence act, and a relaxation function $|_1^+$ for A. If* build-CRPG$(act, A, \mathcal{I}, G, |_1^+)$ *returns FALSE then there is no relaxed plan for $(A, \mathcal{I}, G)$ that starts with the sequence act.*

Proof. Say $act = \langle a_{-n}, \ldots, a_{-1} \rangle$. Say $\langle a_{-n}, \ldots, a_{-1}, a_0, \ldots, a_{m-1} \rangle$ is a relaxed plan for $(A, \mathcal{I}, G)$. We prove below that

$(**)$ if we let the algorithm run until iteration m, then $G \subseteq P(m)$ holds.

Once this is shown, we can argue as follows. If the algorithm returns FALSE then it does so before reaching iteration m, i.e. in iteration $t < m$, with $G \not\subseteq P(t)$. By Lemma 2 all future iterations $t' > t$ have, in particular, $P(t') = P(t)$. So we get $P(m) = P(t)$, in contradiction with $G \subseteq P(m)$.

It remains to prove $(**)$. Denote, for $-n \leqslant t \leqslant m$, by $P|_{rp}(t)$ the set of propositions that are known after $\langle a_{-n}, \ldots, a_{-1}, a_0, \ldots, a_{t-1} \rangle|_1^+$. We show by a simple induction over t that, for all t, $P(t) \supseteq P|_{rp}(t)$. With $t = m$, this suffices because, since $\langle a_{-n}, \ldots, a_{-1}, a_0, \ldots, a_{m-1} \rangle$ is a relaxed plan, the goals are contained in $P|_{rp}(m)$.

Base argument, $t = -n$. Then $\langle a_{-n}, \ldots, a_{-1}, a_0, \ldots, a_{t-1} \rangle$ is empty and the claim is obvious.

Inductive argument, from $t - 1 \Rightarrow t$. First, observe that $a_{t-1} \in A(t-1)$. If $t - 1 < 0$, then this is obvious by construction of the build-CRPG algorithm. If $t - 1 \geqslant 0$ then $a_{t-1} \in A(t-1)$ because $pre(a_{t-1}) \subseteq P|_{rp}(t-1)$ (the actions in the relaxed plan must be applicable), and $P|_{rp}(t-1) \subseteq P(t-1)$ (induction hypothesis). Thus we have that $\langle a_{-n}, \ldots, a_{-1}, a_0, \ldots, a_{t-1} \rangle|_1^+$ is a sub-sequence of $act\text{-}CRPG(t-1)$. But with that, since the relaxation ignores the delete lists, $P|_{rp}(t)$ is a subset of the propositions that are known after $act\text{-}CRPG(t-1)$. The latter set of propositions is, by Lemma 1, equal to $P(t)$. This concludes the argument. □

It is relatively easy to see that the actions selected by the extract-CRPlan procedure do indeed form a relaxed plan for the respective belief state.

Theorem 3 *(CRPG-soundness). Given a conformant task $(A, \mathcal{I}, G)$, an executable action sequence act, and a relaxation function $|_1^+$ for A, s.t.* build-CRPG$(act, A, \mathcal{I}, G, |_1^+)$ *returns TRUE in iteration m. Let $A(0)^s, \ldots, A(m-1)^s$ be the actions selected from $A(0), \ldots, A(m-1)$ by* extract-CRPlan$(CRPG(act, A, \mathcal{I}, G, |_1^+), G)$. *Then $act|_1^+$ concatenated with $A(0)^s, \ldots, A(m-1)^s$ in an arbitrary linearization is a plan for $(A|_1^+, \mathcal{I}, G)$.*

Proof. First, observe that no errors occur in the execution of extract-CRPlan, due to the way the CRPG is computed. During the backwards loop over t, for all $g \in G(t)$, by construction of extract-CRPlan we have $g \in P(t)$. So by construction of build-CRPG either we have $\exists a \in A(t-1), e \in E(a), con(e) \in P(t-1), g \in add(e)$, or we have $\mathcal{I} \rightarrow \bigvee_{l \in Impleafs(g(t))} l$. Also, during the loop, for all actions $a \in A(t)$, by construction of build-CRPG we have that all of a's preconditions are contained in $P(t)$ and thus in some $P(t')$, $t' \leqslant t$.

Denote, for $0 \leqslant t < m$, by $act\text{-}CRPG(t)^s$ the concatenation of $act|_1^+$ with $A(0)^s, \ldots, A(t)^s$ in an arbitrary linearization. We prove that, for any fact g that is made a sub-goal at time t $(0 < t \leqslant m)$ during the execution of extract-CRPlan,

g is known after *act-CRPG*$(t-1)^s$. This shows the claim: the goals are known after *act-CRPG*$(m-1)^s$. The proof proceeds by induction over t. Before we start, note that with $P(t-1) \subseteq P(t)$ for all t (due to the NOOP action), we have that all p that are contained in a set P on which sub-goal is called, but for which $t_0 \leqslant 0$, are contained in $P(0)$. With Lemma 1 we then have that all these p are known after $act|_1^+$.

Base argument, $t = 1$. We distinguish two cases.

- First case, say there exist $a \in A(0), e \in E(a), con(e) \in P(0), g \in add(e)$. Then $a \in A(0)^s$ for one such a and e, and because, as argued above, a's preconditions are known after $act|_1^+$, and $con(e)$ is known after $act|_1^+$, we are done.
- In the other case, actions a were selected from $A(0)$ for all effects e that were responsible for an edge in a path from $l(-n), l \in L$, to $g(1)$, where $\mathcal{I} \to \bigvee_{l \in L} l$. Let I be an initial world state, $I \in 2^{\mathcal{I}}$. Then $l \in I$ for some $l \in L$. There is a path in *Imp* from $l(-n)$ to $g(1)$. For all effects responsible for an edge on that path, the respective action a is in *act-CRPG*$(0)^s$. Because *act* is executable by prerequisite, and because $pre(a) \subseteq P(0)$ for $a \in A(0)$, all actions are applicable at their point of execution in *act-CRPG*$(0)^s$. By construction of *Imp*, and because $l \in I$, it follows that all effects on the path from $l(-n)$ to $g(1)$ occur. So g is true after executing *act-CRPG*$(0)^s$ in I. Since this holds for all $I \in 2^{\mathcal{I}}$, we are done.

Inductive argument, $t-1 \Rightarrow t$. The proof is the same as above for the base argument, except that now we can and must use the induction hypothesis to see that the preconditions of the selected actions a, as well as (if that applies) the effect condition $con(e)$—which are all made sub-goals at layers $t' < t$—will be true at the point of execution of a in *act-CRPG*$(t-1)^s$. In more detail, say g was made a sub-goal at t.

- First case, say there exist $a \in A(t-1), e \in E(a), con(e) \in P(t-1), g \in add(e)$. Then $a \in A(t-1)^s$ for one such a and e. By construction of extract-CRPlan, $pre(a)$ and $con(a)$ were made sub-goals at layers $t' \leqslant t-1$. By induction hypothesis, these propositions are thus known after the respective action sequences *act-CRPG*$(t'-1)^s$. So they are also known after the longer sequence *act-CRPG*$(t-2)^s$. It follows that a is applicable when executed in *act-CRPG*$(t-1)^s$, and that e occurs. So, as required, g is known after *act-CRPG*$(t-1)^s$.
- In the other case, actions a were selected from $A(0), \ldots, A(t-1)$ for all effects e that were responsible for an edge in a path from $l(-n), l \in L$, to $g(t)$, where $\mathcal{I} \to \bigvee_{l \in L} l$. Let $I \in 2^{\mathcal{I}}$, and $l \in I \cap L$. There is a path in *Imp* from $l(-n)$ to $g(t)$. For all effects responsible for an edge on that path, the respective action a is in *act-CRPG*$(t-1)^s$. Say a is selected at time t' ($t' \leqslant t-1$ holds). By construction of extract-CRPlan, a's preconditions were made sub-goals at layers $t'' \leqslant t'$, and, as above in the first case, by induction hypothesis we know that a is applicable when executing *act-CRPG*$(t-1)^s$. With $l \in I$ the effects on the path from $l(-n)$ to $g(t)$ will occur, giving us that g is be true after *act-CRPG*$(0)^s$. This holds for all $I \in 2^{\mathcal{I}}$, and we are done. □

References

[1] B. Bonet, H. Geffner, Planning with incomplete information as heuristic search in belief space, in: S. Chien, R. Kambhampati, C. Knoblock (Eds.), Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS-00), AAAI Press, Menlo Park, CA, 2000, pp. 52–61.

[2] A. Cimatti, M. Roveri, Conformant planning via symbolic model checking, JAIR 13 (2000) 305–338.

[3] P. Bertoli, A. Cimatti, Improving heuristics for planning as search in belief space, in: [30], pp. 143–152.

[4] M. Davis, H. Putnam, A computing procedure for quantification theory, J. ACM 7 (3) (1960) 201–215.

[5] M. Davis, G. Logemann, D. Loveland, A machine program for theorem proving, Comm. ACM 5 (7) (1962) 394–397.

[6] J. Hoffmann, B. Nebel, The FF planning system: Fast plan generation through heuristic search, J. Artificial Intelligence Res. 14 (2001) 253–302.

[7] D. Bryce, S. Kambhampati, D.E. Smith, Planning in belief space with a labelled uncertainty graph, in: Proc. AAAI-04 Workshop on Learning and Planning in Markov Decision Processes, 2004.

[8] P. Ferraris, E. Giunchiglia, Planning as satisfiability in nondeterministic domains, in: Proceedings of the 17th National Conference of the American Association for Artificial Intelligence (AAAI-00), Austin, TX, MIT Press, Cambridge, MA, 2000, pp. 748–753.

[9] E. Giunchiglia, Planning as satisfiability with expressive action languages: Concurrency, constraints, and nondeterminism, in: A. Cohn, F. Giunchiglia, B. Selman (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the 7th International Conference (KR-00), Breckenridge, CO, Morgan Kaufmann, San Mateo, CA, 2000.

[10] J. McCarthy, P.J. Hayes, Some philosophical problems from the standpoint of artificial intelligence, in: B. Meltzer, D. Michie (Eds.), Machine Intelligence, vol. 4, Edinburgh University Press, Edinburgh, UK, 1969, pp. 463–502.

[11] T. Bylander, The computational complexity of propositional STRIPS planning, Artificial Intelligence 69 (1–2) (1994) 165–204.
[12] H. Turner, Polynomial-length planning spans the polynomial hierarchy, in: S. Flesca, S. Greco, N. Leone, G. Ianni (Eds.), Logics in Artificial Intelligence—8th European Conference JELIA-02, Cosenza, Italy, Springer-Verlag, Berlin, 2002, pp. 111–124.
[13] B. Bonet, H. Geffner, Planning as heuristic search, Artificial Intelligence 129 (1–2) (2001) 5–33.
[14] L. Zhang, S. Malik, Extracting small unsatisfiable cores from unsatisfiable boolean formulas, in: Proc. SAT-03, 2003.
[15] A.L. Blum, M.L. Furst, Fast planning through planning graph analysis, Artificial Intelligence 90 (1–2) (1997) 279–298.
[16] M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, S. Malik, Chaff: Engineering an efficient SAT solver, in: Proceedings of the 38th Design Automation Conference (DAC-01), 2001, http://www.ee.princeton.edu/~chaff/DAC2001v56.pdf.
[17] R. Brafman, A simplifier for propositional formulas with many binary clauses, in: [31], pp. 515–520.
[18] R.P.A. Petrick, F. Bacchus, A knowledge-based approach to planning with incomplete information and sensing, in: [30], pp. 212–221.
[19] A Cimatti, M. Roveri, P. Bertoli, Conformant planning via symbolic model checking and heuristic search, Artificial Intelligence 159 (1–2) (2004) 127–206.
[20] J. Slaney, S. Thiebaux, Blocks world revisited, Artificial Intelligence 125 (2001) 119–153.
[21] D.E. Smith, D. Weld, Conformant Graphplan, in: Proceedings of the 15th National Conference of the American Association for Artificial Intelligence (AAAI-98), Madison, WI, MIT Press, Cambridge, MA, 1998, pp. 889–896.
[22] J. Kurien, P.P. Nayak, D.E. Smith, Fragment-based conformant planning, in: [30], pp. 153–162.
[23] A. Cimatti, M. Roveri, P. Bertoli, Heuristic search + symbolic model checking = efficient conformant planning, in: [31], pp. 467–472.
[24] D. Bryce, S. Kambhampari, Heuristic guidance measures for conformant planning, in: [29], pp. 365–374.
[25] J. Rintanen, Constructing conditional plans by a theorem-prover, J. Artificial Intelligence Res. 10 (1999) 323–352.
[26] T. Eiter, W. Faber, N. Leone, G. Pfeifer, A logic programming approach to knowledge-state planning, II: The DLVK system, Artificial Intelligence 144 (1–2) (2003) 157–211.
[27] B.C. Gazen, C. Knoblock, Combining the expressiveness of UCPOP with the efficiency of Graphplan, in: S. Steel, R. Alami (Eds.), Recent Advances in AI Planning. 4th European Conference on Planning (ECP'97), in: Lecture Notes in Artificial Intelligence, vol. 1348, Toulouse, France, Springer-Verlag, Berlin, 1997, pp. 221–233.
[28] J. Hoffmann, R. Brafman, Contingent planning via heuristic forward search with implicit belief states, in: S. Biundo, K. Myers, K. Rajan (Eds.), Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05), Monterey, CA, Morgan Kaufmann, San Mateo, CA, 2005, in press.
[29] S. Koenig, S. Zilberstein, J. Koehler (Eds.), Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04), Whistler, Canada, Morgan Kaufmann, San Mateo, CA, 2004.
[30] M. Ghallab, J. Hertzberg, P. Traverso (Eds.), Proceedings of the 6th International Conference on Artificial Intelligence Planning and Scheduling (AIPS-02), Toulouse, France, Morgan Kaufmann, San Mateo, CA, 2002.
[31] B. Nebel (Ed.), Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), Seattle, Washington, Morgan Kaufmann, San Mateo, CA, 2001.

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Artificial Intelligence 170 (2006) 542–580
Artificial Intelligence
www.elsevier.com/locate/artint

Causes and explanations in the structural-model approach: Tractable cases ☆

Thomas Eiter [a], Thomas Lukasiewicz [b,a,*]

[a] *Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, 1040 Vienna, Austria*
[b] *Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Rome, Italy*

Received 19 August 2004; received in revised form 25 November 2005; accepted 21 December 2005

Available online 8 February 2006

Abstract

This paper continues the research on the computational aspects of Halpern and Pearl's causes and explanations in the structural-model approach. To this end, we first explore how an instance of deciding weak cause can be reduced to an equivalent instance in which irrelevant variables in the (potential) weak cause and the causal model are removed, which extends previous work by Hopkins. We then present a new characterization of weak cause for a certain class of causal models in which the causal graph over the endogenous variables has the form of a directed chain of causal subgraphs, called *decomposable causal graph*. Furthermore, we also identify two important subclasses in which the causal graph over the endogenous variables forms a directed tree and more generally a directed chain of layers, called *causal tree* and *layered causal graph*, respectively. By combining the removal of irrelevant variables with this new characterization of weak cause, we then obtain techniques for deciding and computing causes and explanations in the structural-model approach, which can be done in polynomial time under suitable restrictions. This way, we obtain several tractability results for causes and explanations in the structural-model approach. To our knowledge, these are the first explicit ones. They are especially useful for dealing with structure-based causes and explanations in first-order reasoning about actions, which produces large causal models that are naturally layered through the time line, and thus have the structure of layered causal graphs. Furthermore, an important feature of the tractable cases for causal trees and layered causal graphs is that they can be recognized efficiently, namely in linear time. Finally, by extending the new characterization of weak cause, we obtain similar techniques for computing the degrees of responsibility and blame, and hence also novel tractability results for structure-based responsibility and blame.

Keywords: Structural causal model; Probabilistic structural causal model; Weak cause; Actual cause; Explanation; α-partial explanation; Partial explanation; Explanatory power; Responsibility; Blame; Computational complexity; Tractability

☆ This paper is a significantly extended and revised version of a paper that appeared in: Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI-2002), August 1–4, 2002, Edmonton, Alberta, Canada, Morgan Kaufmann, 2002, pp. 146–153.

* Corresponding author.
E-mail addresses: eiter@kr.tuwien.ac.at (T. Eiter), lukasiewicz@dis.uniroma1.it (T. Lukasiewicz).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2005.12.003

1. Introduction

Dealing with causality is an important issue which emerges in many applications of AI. The existing approaches to causality in AI can be roughly divided into those that have been developed as modal nonmonotonic logics (especially in logic programming) and those that evolved from the area of Bayesian networks. A representative of the former is Geffner's modal nonmonotonic logic for handling causal knowledge [12,13], which is inspired by default reasoning from conditional knowledge bases. Other modal-logic based formalisms play an important role in dealing with causal knowledge about actions and change; see especially the work by Turner [36] and the references therein for an overview. A representative of the latter is Pearl's approach to modeling causality by structural equations [1,10,30,31], which is central to a number of recent research efforts. In particular, the evaluation of deterministic and probabilistic counterfactuals has been explored, which is at the core of problems in fault diagnosis, planning, decision making, and determination of liability [1]. It has been shown that the structural-model approach allows a precise modeling of many important causal relationships, which can especially be used in natural language processing [10]. An axiomatization of reasoning about causal formulas in the structural-model approach has been given by Halpern [14].

Causality also plays an important role in the generation of explanations, which are of crucial importance in areas like planning, diagnosis, natural language processing, and probabilistic inference. Different notions of explanations have been studied quite extensively, see especially [11,19,34] for philosophical work, and [20,29,35] for work in AI related to Bayesian networks. A critical examination of such approaches from the viewpoint of explanations in probabilistic systems is given in [2].

In [15], Halpern and Pearl formalized causality using a model-based definition, which allows for a precise modeling of many important causal relationships. Based on a notion of weak causality, they offer appealing definitions of actual causality [16] and causal explanations [18]. As they show, their notions of actual cause and causal explanation, which is very different from the concept of causal explanation in [12,26,27], models well many problematic examples in the literature.

The following example from [3] illustrates the structural-model approach. Roughly, structural causal models consist of a set of random variables, which may have a causal influence on each other. The variables are divided into exogenous variables, which are influenced by factors outside the model, and endogenous variables, which are influenced by exogenous and endogenous variables. This latter influence is described by structural equations for the endogenous variables. For more details on structural causal models, we refer to Section 2 and especially to [1,10,14,30,31].

Example 1.1 *(rock throwing).* Suppose that Suzy and Billy pick up rocks and throw them at a bottle. Suzy's rock gets there first, shattering the bottle. Since both throws are fully accurate, Billy's rock would have shattered the bottle, if Suzy had not thrown. We may model such a scenario in the structural-model approach as follows. We assume two binary background variables U_S and U_B, which determine the motivation and the state of mind of Suzy and Billy, where U_S (resp., U_B) is 1 iff Suzy (resp., Billy) intends to throw a rock. We then have five binary variables ST, BT, SH, BH, and BS, which describe the observable situation, where ST (resp., BT) is 1 iff Suzy (resp., Billy) throws a rock, SH (resp., BH) is 1 iff Suzy's (resp., Billy's) rock hits the bottle, and BS is 1 iff the bottle shatters. The causal dependencies between these variables are expressed by functions, which say that (i) the value of ST (resp., BT) is given by the value of U_S (resp., U_B), (ii) SH is 1 iff ST is 1, (iii) BH is 1 iff BT is 1 and SH is 0, and (iv) BS is 1 iff SH or BH is 1. These dependencies can be graphically represented as in Fig. 1.

Some actual causes and explanations in the structural-model approach are then informally given as follows. If both Suzy and Billy intend to throw a rock, then (i) Suzy's throwing a rock is an *actual cause* of the bottle shattering, while (ii) Billy's throwing a rock is not. Furthermore, (iii) if either Suzy or Billy intends to throw a rock, then Suzy's throwing a rock is an *explanation* of the bottle shattering. Here, (i)–(iii) are roughly determined as follows. As for (i), if both Suzy and Billy intend to throw a rock, then Suzy actually throws a rock, and the bottle actually shatters.

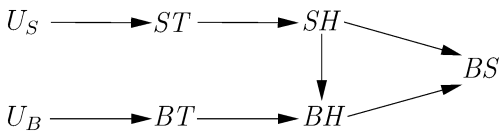

Fig. 1. Causal graph.

Moreover, under the *structural contingency* that Billy does not throw a rock, (a) if Suzy does not throw a rock, then the bottle does not shatter, and (b) if Suzy throws a rock, then the bottle shatters, even if any of the other variables would take their actual values. As for (ii), there is no structural contingency under which (a) if Billy does not throw a rock, then the bottle does not shatter, and (b) if Billy throws a rock, then the bottle shatters, even if any of the other variables would take their actual values. Finally, as for (iii), if either Suzy or Billy intends to throw a rock, then the bottle actually shatters, Suzy's throwing a rock is a cause of the bottle shattering whenever she actually throws a rock, and there are some possible contexts in which Suzy throws a rock and some in which she does not. Intuitively, there should be a possible context in which the explanation is false, so that it is not already known, and a possible context in which the explanation is true, so that it is not vacuous.

There are a number of recent papers that are based on Halpern and Pearl's definitions of actual causality [16] and causal explanations [18]. In particular, Chockler and Halpern [3] define the notions of responsibility and blame as a refinement of actual causality. Chockler, Halpern, and Kupferman [4] then make use of the notion of responsibility for verifying a system against a formal specification. Along another line of application, Hopkins and Pearl [23] and Finzi and Lukasiewicz [9] generalize structure-based causes and explanations to a first-order framework and make them available in situation-calculus-based reasoning about actions (see Section 3). Furthermore, Hopkins and Pearl [24] explore the usage of structure-based causality [16] for commonsense causal reasoning. Finally, inspired by Halpern and Pearl's notions of actual causality [16] and causal explanations [18], Park [28] presents a novel approach allowing for different causal criteria that are influenced by psychological factors not representable in a structural causal model.

The semantic aspects of causes and explanations in the structural-model approach have been thoroughly studied in [15–18]. Their computational complexity has been analyzed in [6,7], where it has been shown that associated decision problems are intractable in general. For example, deciding actual causes (as defined in [16]) is complete for the class Σ_2^{P} $(= \mathrm{NP}^{\mathrm{NP}})$ of the Polynomial Hierarchy, while deciding whether an explanation over certain variables exists is complete for Σ_3^{P} $(= \mathrm{NP}^{\Sigma_2^{\mathrm{P}}})$. Thus, these problems are "harder" than the classical propositional satisfiability problem (which is NP-complete), but "easier" than PSPACE-complete problems. Chockler and Halpern [3] and Chockler, Halpern, and Kupferman [4] have shown that computing the degrees of responsibility and blame is complete for polynomial time computation with restricted use of a Σ_2^{P} oracle (see Section 4.4). As for algorithms, Hopkins [21] explored search-based strategies for computing actual causes in both the general and restricted settings.

However, to our knowledge, no tractable cases for causes and explanations in the structural-model approach were explicitly known so far. In this paper, we aim at filling this gap and provide non-trivial tractability results for the main computational problems on causes and explanations. These tractability results are especially useful for dealing with structure-based causes and explanations in first-order reasoning about actions as recently introduced in [9], where one has to handle binary causal models with a quite large number of variables (see Section 3). We make contributions to several issues, which are briefly summarized as follows:

- The first issue concerns focusing of the computation to the relevant part of the causal model. Extending work by Hopkins [21], we explore how an instance of deciding weak cause can be reduced to an equivalent instance in which the (potential) weak cause and the causal model may contain fewer variables. That is, irrelevant variables in weak causes and causal models are identified and removed. We provide two such reductions in this paper, which have different properties, but can be both carried out in polynomial time. These reductions can lead to great simplifications in (potential) weak causes and causal models, and thus speed up considerably computations about causes and explanations. Notice that weak causes are fundamental to the notion of actual cause, to various forms of explanations, as well as to the notions of responsibility and blame.
- The second issue to which we contribute are characterizations of weak causes in the structural-model approach. We present a novel such characterization for a class of causal models in which the causal graph over the endogenous variables has the form of a directed chain of causal subgraphs, which we call a *decomposable causal graph*. We also identify two natural subclasses of decomposable causal graphs, where the causal graph over the endogenous variables forms a directed tree and, more generally, a directed chain of layers, which we call a *causal tree* and a *layered causal graph*, respectively, and provide simplified versions of the characterizations of weak causes.
- By combining the removal of irrelevant variables (in weak causes and causal models) with this new characterization of weak cause in the above causal models, we obtain algorithms for deciding and computing weak causes, actual causes, explanations, partial explanations, and α-partial explanations, as well as for computing the ex-

planatory power of partial explanations, which all run in polynomial time under suitable conditions. This way, we obtain several tractability results for the structural-model approach. To our knowledge, these are the first ones that are explicitly derived for structure-based causes and explanations.

- Furthermore, by slightly extending the new characterization of weak cause in the above causal models, we also obtain algorithms for computing the degrees of responsibility and blame in the structural-model approach, which similarly run in polynomial time under suitable conditions. We thus also obtain new tractability results for the structure-based notions of responsibility and blame. Note that Chockler, Halpern, and Kupferman [4] have recently shown that computing the degree of responsibility in read-once Boolean formulas (which are Boolean formulas in which each variable occurs at most once) is possible in linear time.
- The tractability results for layered causal graphs are especially useful for dealing with structure-based causes and explanations in first-order reasoning about actions [9], which produces large binary causal models that have a natural layering through the time line and thus the structure of layered causal graphs (see Section 3). Finally, we also show that all the above techniques and results carry over to a generalization of causal models to *extended causal models*, which has been recently introduced by Halpern and Pearl in [17].

An attractive feature of the tractable cases identified for causal trees and layered causal graphs is that the respective problem instances can be recognized efficiently, namely in linear time. For general decomposable causal graphs, however, this is not the case, since this problem is NP-complete in general. Nonetheless, effort spent for the recognition may be more than compensated by the speed up in solving the reasoning problems on weak causes and explanations.

Our results on the computational and semantic properties of weak causes and explanations help, as we believe, to enlarge the understanding of and insight into the structural-model approach by Halpern and Pearl and its properties. Furthermore, they provide the basis for developing efficient algorithms and pave the way for implementations. For example, complexity results on answer set programming [5] have guided the development of efficient solvers such as DLV [25]. The results of this paper are in particular of interest and significant, since a structural decomposition seems natural and applies to a number of examples from the literature.

The rest of this paper is organized as follows. Section 2 contains some preliminaries on structural causal models as well as on causes, explanations, responsibility, and blame in structural causal models. Section 3 describes how structure-based causal concepts can be defined in first-order reasoning about actions. In Section 4, we describe the decision and optimization problems for which we present tractability results in this paper, and we summarize previous complexity results for these problems. In Section 5, we explore the removal of irrelevant variables when deciding weak cause. Section 6 presents tractability results for causal trees. Section 7 then generalizes to decomposable causal graphs, while Section 8 concentrates on layered causal graphs. In Section 9, we generalize the above techniques and results to extended causal models. Section 10 summarizes our results and gives a conclusion. To increase readability, all proofs have been moved to Appendices A–E.

2. Preliminaries

In this section, we give some technical preliminaries. We recall Pearl's structural causal models and Halpern and Pearl's notions of weak and actual cause [15–17] and their notions of explanation, partial explanation, and explanatory power [15,18].

2.1. Causal models

We start with recalling structural causal models; for further background and motivation, see especially [1,10,14,30, 31]. Roughly, the main idea behind structural causal models is that the world is modeled by random variables, which may have a causal influence on each other. The variables are divided into exogenous variables, which are influenced by factors outside the model, and endogenous variables, which are influenced by exogenous and endogenous variables. This latter influence is described by structural equations for the endogenous variables.

More formally, we assume a finite set of *random variables*. Capital letters U, V, W, etc. denote variables and sets of variables. Each variable X_i may take on *values* from a finite *domain* $D(X_i)$. A *value* for a set of variables $X = \{X_1, \ldots, X_n\}$ is a mapping $x : X \rightarrow D(X_1) \cup \cdots \cup D(X_n)$ such that $x(X_i) \in D(X_i)$ for all $i \in \{1, \ldots, n\}$; for $X = \emptyset$, the unique value is the empty mapping $\emptyset$. The *domain* of X, denoted $D(X)$, is the set of all values for X.

We say that X is *domain-bounded* iff $|D(X_i)| \leqslant k$ for every $X_i \in X$, where k is some global constant. Lower case letters x, y, z, etc. denote values for the variables or the sets of variables X, Y, Z, etc., respectively. Assignments $X = x$ of values to variables are often abbreviated by the value x. We often identify singletons $\{X_i\}$ with X_i, and their values x with $x(X_i)$.

For $Y \subseteq X$ and $x \in D(X)$, we denote by $x|Y$ the restriction of x to Y. For disjoint sets of variables X, Y and values $x \in D(X)$, $y \in D(Y)$, we denote by xy the union of x and y. For (not necessarily disjoint) sets of variables X, Y and values $x \in D(X)$, $y \in D(Y)$, we denote by $[x\langle y]$ the union of $x|(X \setminus Y)$ and y.

A *causal model* $M = (U, V, F)$ consists of two disjoint finite sets U and V of *exogenous* and *endogenous* variables, respectively, and a set $F = \{F_X \mid X \in V\}$ of functions that assign a value of X to each value of the *parents* $PA_X \subseteq U \cup V \setminus \{X\}$ of X. Every value $u \in D(U)$ is also called a *context*. We call a causal model $M = (U, V, F)$ *domain-bounded* iff every $X \in V$ is domain-bounded. In particular, M is *binary* iff $|D(X)| = 2$ for all $X \in V$. The parent relationship between the variables of $M = (U, V, F)$ is expressed by the *causal graph* for M, denoted $G(M)$, which is the directed graph (N, E) that has $U \cup V$ as the set of nodes N, and a directed edge from X to Y in E iff X is a parent of Y, for all variables $X, Y \in U \cup V$. We use $G_V(M)$ to denote the subgraph of $G(M)$ induced by V.

We focus here on the principal class of *recursive* causal models $M = (U, V, F)$; as argued in [15], we do not lose much generality by concentrating on recursive causal models. A causal model $M = (U, V, F)$ is *recursive*, if its causal graph is a directed acyclic graph. Equivalently, there exists a total ordering $\prec$ on V such that $Y \in PA_X$ implies $Y \prec X$, for all $X, Y \in V$. In recursive causal models, every assignment $U = u$ to the exogenous variables determines a unique value y for every set of endogenous variables $Y \subseteq V$, denoted by $Y_M(u)$ (or simply by $Y(u)$, if M is understood). In the following, M is reserved for denoting a recursive causal model.

Example 2.1 *(rock throwing cont'd).* The causal model $M = (U, V, F)$ for Example 1.1 is given by $U = \{U_S, U_B\}$, $V = \{ST, BT, SH, BH, BS\}$, and $F = \{F_{ST}, F_{BT}, F_{SH}, F_{BH}, F_{BS}\}$, where $F_{ST} = U_S$, $F_{BT} = U_B$, $F_{SH} = ST$, $F_{BH} = 1$ iff $BT = 1$ and $SH = 0$, and $F_{BS} = 1$ iff $SH = 1$ or $BH = 1$. Fig. 1 shows the causal graph for M, that is, the parent relationships between the exogenous and endogenous variables in M. Since this graph is acyclic, M is recursive.

In a causal model, we may set endogenous variables X to a value x by an "external action". More formally, for any causal model $M = (U, V, F)$, set of endogenous variables $X \subseteq V$, and value $x \in D(X)$, the causal model $M_{X=x}$ is given by $(U, V \setminus X, F_{X=x})$, where $F_{X=x} = \{F'_Y \mid Y \in V \setminus X\}$ and each F'_Y is obtained from F_Y by setting X to x, is a *submodel* of M. We use M_x and F_x to abbreviate $M_{X=x}$ and $F_{X=x}$, respectively, if X is understood from the context. Similarly, for a set of endogenous variables $Y \subseteq V$ and $u \in D(U)$, we write $Y_x(u)$ to abbreviate $Y_{M_x}(u)$. We assume that $X(u) = x$ for all $u \in D(U)$ in the submodel of M where X is set to x.

As for computation, we assume for causal models $M = (U, V, F)$ no particular form of representation of the functions $F_X : D(PA_X) \to D(X)$, $X \in V$, in F (by formulas, circuits, etc.), but that every F_X is evaluable in polynomial time. Furthermore, we assume that the causal graph $G(M)$ for M is part of the input representation of M. Notice that $G(M)$ is computable from M with any common representation of the functions F_X (by formulas, circuits, etc.) in time linear in the size of the representation of M anyway. For any causal model M, we denote by $\|M\|$ the size of its representation.

The following proposition is then immediate.

Proposition 2.1. *For all $X, Y \subseteq V$ and $x \in D(X)$, the values $Y(u)$ and $Y_x(u)$, given $u \in D(U)$, are computable in polynomial time.*

2.2. Weak and actual causes

We now recall weak and actual causes from [15–17]. We first define events and the truth of events in a causal model $M = (U, V, F)$ under a context $u \in D(U)$.

A *primitive event* is an expression of the form $Y = y$, where Y is an endogenous variable and y is a value for Y. The set of *events* is the closure of the set of primitive events under the Boolean operations $\neg$ and $\wedge$ (that is, every primitive event is an event, and if ϕ and ψ are events, then also $\neg\phi$ and $\phi \wedge \psi$). For any event ϕ, we denote by $V(\phi)$ the set of all variables in ϕ.

The *truth* of an event ϕ in a causal model $M = (U, V, F)$ under a context $u \in D(U)$, denoted $(M, u) \models \phi$, is inductively defined as follows:

- $(M, u) \models Y = y$ iff $Y_M(u) = y$;
- $(M, u) \models \neg\phi$ iff $(M, u) \models \phi$ does not hold;
- $(M, u) \models \phi \wedge \psi$ iff $(M, u) \models \phi$ and $(M, u) \models \psi$.

Further operators $\vee$ and $\rightarrow$ are defined as usual, that is, $\phi \vee \psi$ and $\phi \rightarrow \psi$ stand for $\neg(\neg\phi \wedge \neg\psi)$ and $\neg\phi \vee \psi$, respectively. We write $\phi_M(u)$ (resp., $\phi(u)$ if M is understood) to abbreviate $(M, u) \models \phi$. For $X \subseteq V$ and $x \in D(X)$, we use $\phi_{M_x}(u)$ (resp., $\phi_x(u)$) as an abbreviation of $(M_x, u) \models \phi$. For $X = \{X_1, \ldots, X_k\} \subseteq V$ with $k \geqslant 1$ and $x_i \in D(X_i)$, $1 \leqslant i \leqslant k$, we use $X = x_1 \cdots x_k$ to abbreviate the event $X_1 = x_1 \wedge \cdots \wedge X_k = x_k$. For any event ϕ, we denote by $\|\phi\|$ its size, which is the number of symbols in it.

The following result follows immediately from Proposition 2.1.

Proposition 2.2. *Let $X \subseteq V$ and $x \in D(X)$. Given $u \in D(U)$ and an event ϕ, deciding whether $\phi(u)$ holds (resp., $\phi_x(u)$ holds for given x) is feasible in polynomial time.*

We are now ready to recall the notions of weak and actual cause as recently introduced by Halpern and Pearl in [17]. Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$ and $x \in D(X)$, and let ϕ be an event. Then, $X = x$ is a *weak cause* of ϕ under $u \in D(U)$ iff the following conditions hold:

AC1. $X(u) = x$ and $\phi(u)$.
AC2. Some $W \subseteq V \setminus X$ and some $\bar{x} \in D(X)$ and $w \in D(W)$ exist such that:
(a) $\neg\phi_{\bar{x}w}(u)$, and
(b) $\phi_{xw'\hat{z}}(u)$ for all $W' \subseteq W$, $\hat{Z} \subseteq V \setminus (X \cup W)$, $w' = w|W'$, and $\hat{z} = \hat{Z}(u)$.

Loosely speaking, **AC1** says that both $X = x$ and ϕ hold under u, while **AC2** expresses that $X = x$ is a nontrivial reason for ϕ. Here, the dependence of ϕ from $X = x$ is tested under special *structural contingencies*, where some $W \subseteq V \setminus X$ is kept at some value $w \in D(W)$. **AC2**(a) says that ϕ can be false for other values of X under w, while **AC2**(b) essentially ensures that X alone is sufficient for the change from ϕ to $\neg\phi$. Observe that $X = x$ can be a weak cause only if X is nonempty.

Furthermore, $X = x$ is an *actual cause* of ϕ under u iff additionally the following minimality condition is satisfied:

AC3. X is minimal. That is, no proper subset of X satisfies both **AC1** and **AC2**.

The following result says that for singletons X, it holds that $X = x$ is a weak cause of ϕ under u iff $X = x$ is an actual cause of ϕ under u. This result is immediate by the observation that X is nonempty for every weak cause $X = x$ of ϕ under u.

Theorem 2.3. *Let $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, and $u \in D(U)$. Let ϕ be an event. If X is a singleton, then $X = x$ is a weak cause of ϕ under u iff $X = x$ is an actual cause of ϕ under u.*

We give an example to illustrate the above notions of weak and actual cause.

Example 2.2 *(rock throwing cont'd).* Consider the context $u_{1,1} = (1, 1)$ in which both Suzy and Billy intend to throw a rock. Then, both $ST = 1$ and $ST = 1 \wedge BT = 1$ are weak causes of $BS = 1$, while $BT = 1$ is not. For instance, let us show that $ST = 1$ is a weak cause of $BS = 1$ under $u_{1,1}$. As for **AC1**, both ST and BS are 1 under $u_{1,1}$. As for **AC2**, under the contingency that BT is set to 0, we have that (a) if ST is set to 0, then BS has the value 0, and (b) if ST is set to 1, then BS is 1. In fact, $ST = 1$ is an actual cause of $BS = 1$ under $u_{1,1}$ (by Theorem 2.3), while $ST = 1 \wedge BT = 1$ is not. Furthermore, $ST = 1$ (resp., $BT = 1$) is a weak (and by Theorem 2.3 also an actual) cause of $BS = 1$ under $u_{1,0} = (1, 0)$ (resp., $u_{0,1} = (0, 1)$) in which only Suzy (resp., Billy) intends to throw a rock.

Note that the above notion of weak cause from [17] slightly refines the notion of weak cause from [15,16], which uses the following **AC2′** instead of **AC2**:

AC2′. Some $W \subseteq V \setminus X$ and some $\overline{x} \in D(X)$ and $w \in D(W)$ exist such that:
(a) $\neg\phi_{\overline{x}w}(u)$, and
(b) $\phi_{xw\hat{z}}(u)$ for all $\hat{Z} \subseteq V \setminus (X \cup W)$ and $\hat{z} = \hat{Z}(u)$.

Here, only $\phi_{xw\hat{z}}(u)$ holds in (b), but not necessarily $\phi_{xw'\hat{z}}(u)$ for all $W' \subseteq W$ and $w' = w|W$. However, as pointed out by Hopkins and Pearl [24], this earlier weak cause is too permissive, which is shown by the following example from [24].

Example 2.3. Suppose that a prisoner dies either if a loads b's gun and b shoots, or if c loads and shoots his gun. We assume the binary endogenous variables A, B, C, and D, where (i) $A = 1$ iff a loads b's gun, (ii) $B = 1$ iff b shoots, (iii) $C = 1$ iff c loads and shoots his gun, and (iv) $D = 1$ iff the prisoner dies, that is, iff $(A = 1 \wedge B = 1) \vee (C = 1)$. Consider the context u in which a loads b's gun, b does not shoot, and c loads and shoots his gun, and thus the prisoner dies. Then, $C = 1$ should be a weak cause of $D = 1$, which is in fact the case under both definitions. Furthermore, $A = 1$ should not be a weak cause of $D = 1$. However, this is only the case for the notion of weak cause from [17], but not for the one from [15,16].

We note that all results of this paper also carry over to the earlier weak cause [8].

2.3. Explanations

We next recall the concept of an explanation from [15,18]. Intuitively, an explanation of an observed event ϕ is a minimal conjunction of primitive events that causes ϕ even when there is uncertainty about the actual situation at hand. The agent's epistemic state is given by a set of possible contexts $u \in D(U)$, which describes all the possible scenarios for the actual situation.

Formally, let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$ and $x \in D(X)$, let ϕ be an event, and let $\mathcal{C} \subseteq D(U)$ be a set of contexts. Then, $X = x$ is an *explanation* of ϕ relative to $\mathcal{C}$ iff the following conditions hold:

EX1. $\phi(u)$ holds, for every context $u \in \mathcal{C}$.
EX2. $X = x$ is a weak cause of ϕ under every $u \in \mathcal{C}$ such that $X(u) = x$.
EX3. X is minimal. That is, for every $X' \subset X$, some context $u \in \mathcal{C}$ exists such that $X'(u) = x|X'$ and $X' = x|X'$ is not a weak cause of ϕ under u.
EX4. $X(u) = x$ and $X(u') \neq x$ for some $u, u' \in \mathcal{C}$.

Note that in **EX3**, any counterexample $X' \subset X$ to minimality must be a nonempty set of variables. The following example illustrates the above notion of explanation.

Example 2.4 *(rock throwing cont'd).* Consider the set of contexts $\mathcal{C} = \{u_{1,1}, u_{1,0}, u_{0,1}\}$. Then, $ST = 1$ is an explanation of $BS = 1$ relative to $\mathcal{C}$, since **EX1** $BS(u_{1,1}) = BS(u_{1,0}) = BS(u_{0,1}) = 1$, **EX2** $ST = 1$ is a weak cause of $BS = 1$ under both $u_{1,1}$ and $u_{1,0}$, **EX3** ST is obviously minimal, and **EX4** $ST(u_{1,1}) = 1$ and $ST(u_{0,1}) \neq 1$. Furthermore, $ST = 1 \wedge BT = 1$ is not an explanation of $BS = 1$ relative to $\mathcal{C}$, since here the minimality condition **EX3** is violated.

2.4. Partial explanations and explanatory power

We next recall the notions of α-partial/partial explanations and of explanatory power of partial explanations [15, 18]. Roughly, the main idea behind partial explanations is to generalize the notion of explanation of Section 2.3 to a setting where additionally a probability distribution over the set of possible contexts is given.

Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$ and $x \in D(X)$. Let ϕ be an event, and let $\mathcal{C} \subseteq D(U)$ be such that $\phi(u)$ for all $u \in \mathcal{C}$. We use $\mathcal{C}^{\phi}_{X=x}$ to denote the largest subset $\mathcal{C}'$ of $\mathcal{C}$ such that $X = x$ is an explanation of ϕ relative

to $\mathcal{C}'$. Note that this set $\mathcal{C}^{\phi}_{X=x}$ is unique. The following proposition from [7] shows that $\mathcal{C}^{\phi}_{X=x}$ is defined, if a subset $\mathcal{C}'$ of $\mathcal{C}$ exists such that $X = x$ is an explanation of ϕ relative to $\mathcal{C}'$; it also characterizes $\mathcal{C}^{\phi}_{X=x}$.

Proposition 2.4. *Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$ and $x \in D(X)$. Let ϕ be an event, and let $\mathcal{C} \subseteq D(U)$ be such that $\phi(u)$ for all $u \in \mathcal{C}$. If $X = x$ is an explanation of ϕ relative to some $\mathcal{C}' \subseteq \mathcal{C}$, then $\mathcal{C}^{\phi}_{X=x}$ is the set of all $u \in \mathcal{C}$ such that either* (i) $X(u) \neq x$, *or* (ii) $X(u) = x$ *and $X = x$ is a weak cause of ϕ under u.*

Let P be a probability function on $\mathcal{C}$ (that is, P is a mapping from $\mathcal{C}$ to the interval $[0, 1]$ such that $\sum_{u\in\mathcal{C}} P(u) = 1$), and define

$$P\big(\mathcal{C}^{\phi}_{X=x} \mid X = x\big) = \sum_{\substack{u\in\mathcal{C}^{\phi}_{X=x},\\ X(u)=x}} P(u) \Big/ \sum_{\substack{u\in\mathcal{C},\\ X(u)=x}} P(u).$$

Then, $X = x$ is called an *α-partial explanation* of ϕ relative to $(\mathcal{C}, P)$ iff $\mathcal{C}^{\phi}_{X=x}$ is defined and $P(\mathcal{C}^{\phi}_{X=x} \mid X = x) \geqslant \alpha$. We say $X = x$ is a *partial explanation* of ϕ relative to $(\mathcal{C}, P)$ iff $X = x$ is an α-partial explanation of ϕ relative to $(\mathcal{C}, P)$ for some $\alpha > 0$; furthermore, $P(\mathcal{C}^{\phi}_{X=x} \mid X = x)$ is called its *explanatory power* (or *goodness*).

Example 2.5 *(rock throwing cont'd).* Consider the set of contexts $\mathcal{C} = \{u_{1,1}, u_{1,0}, u_{0,1}\}$, and let $P(u_{1,1}) = 0.2$ and $P(u_{1,0}) = P(u_{0,1}) = 0.4$. Then, $\mathcal{C}^{BS=1}_{ST=1} = \mathcal{C}$, and thus $ST = 1$ is a 1-partial explanation of $BS = 1$ relative to $(\mathcal{C}, P)$. That is, $ST = 1$ is a partial explanation of $BS = 1$ relative to $(\mathcal{C}, P)$ with explanatory power 1.

As for computation, we assume that the above probability functions P on $\mathcal{C}$ are computable in polynomial time.

2.5. Responsibility and blame

We finally recall the notions of responsibility and blame from [3]. Intuitively, the notion of responsibility is a refinement of the notion of actual cause, which also measures the minimal number of changes that must be made under a structural contingency to create a counterfactual dependence of ϕ from $X = x$. Whereas the notion of blame then also takes into consideration the belief of an agent about the possible causal models and contexts (before setting the weak cause).

In the sequel, let $M = (U, V, F)$ be a causal model, let $X \subseteq V$, $x \in D(X)$, and $u \in D(U)$, and let ϕ be an event. Let us call the pair (M, u) a *situation*. Then, the *degree of responsibility* of $X = x$ for ϕ in situation (M, u), denoted $\mathrm{dr}((M, u), X = x, \phi)$, is 0 if $X = x$ is not an actual cause of ϕ under u in M, and it is $1/(k+1)$ if $X = x$ is an actual cause of ϕ under u in M, and

(i) some $W \subseteq V \setminus X$, $\overline{x} \in D(X)$, and $w \in D(W)$ exist such that **AC2**(a) and (b) hold and that k variables in W have different values in w and $W(u)$, and
(ii) no $\overline{W} \subseteq V \setminus X$, $\overline{x}' \in D(X)$, and $\overline{w} \in D(\overline{W})$ exist such that **AC2**(a) and (b) hold and that $k' < k$ variables in $\overline{W}$ have different values in $\overline{w}$ and $\overline{W}(u)$.

Informally, $\mathrm{dr}((M, u), X = x, \phi) = 1/(k+1)$, where k is the minimal number of changes that have to be made under u in M to make ϕ counterfactually depend on $X = x$. In particular, if $X = x$ is not an actual cause of ϕ under u in M, then $k = \infty$, and thus $\mathrm{dr}((M, u), X = x, \phi) = 0$. Otherwise, $\mathrm{dr}((M, u), X = x, \phi)$ is at most 1.

Example 2.6 *(rock throwing cont'd).* Consider again the context $u_{1,1} = (1, 1)$ in which both Suzy and Billy intend to throw a rock. As argued in Example 2.2, Suzy's throwing a rock ($ST = 1$) is an actual cause of the bottle shattering ($BS = 1$), witnessed by the contingency that Billy does not throw (and hence does not hit). Here, **AC2** holds also under the contingency that Billy throws a rock, but the rock does not hit the bottle (BT and BH are set to 1 and 0, respectively). Since BT and BH are 1 and 0, respectively, under $u_{1,1}$, the degree of responsibility of Suzy's throwing a rock ($ST = 1$) for the bottle shattering ($BS = 1$) in $(M, u_{1,1})$ is given by 1.

An *epistemic state* $\mathcal{E} = (\mathcal{K}, P)$ consists of a set of situations $\mathcal{K}$ and a probability distribution P over $\mathcal{K}$. The *degree of blame* of setting X to x for ϕ relative to an epistemic state $(\mathcal{K}, P)$, denoted $\mathrm{db}(\mathcal{K}, P, X \leftarrow x, \phi)$, is defined as

$$\sum_{(M,u)\in\mathcal{K}} \mathrm{dr}\big((M_{X=x}, u), X = x, \phi\big) \cdot P\big((M, u)\big).$$

Informally, $(\mathcal{K}, P)$ are the situations that an agent considers possible before X is set to x along with their probabilities believed by the agent. Then, $\mathrm{db}(\mathcal{K}, P, X \leftarrow x, \phi)$ is the expected degree of responsibility of $X = x$ for ϕ in $(M_{X=x}, u)$.

Example 2.7 *(rock throwing cont'd).* Suppose that we are computing the degree of blame of Suzy's throwing a rock for the bottle shattering. Assume that Suzy considers possible a modified version of the causal model given in Example 2.1, denoted M', where Billy may also throw extra hard, which is expressed by the additional value 2 of U_B and BT. If Billy throws extra hard, then Billy's rock hits the bottle independently of what Suzy does, which is expressed by additionally assuming that BH is 1 if BT is 2. Assume then that Suzy considers possible the contexts $u_{1,0}$, $u_{1,1}$, and $u_{1,2}$, where Suzy throws a rock, and Billy either does not throw a rock, throws a rock in a normal way, or throws a rock extra hard. Finally, assume that each of the three contexts has the probability $1/3$. It is then not difficult to verify that the degree of responsibility of Suzy's throwing a rock for the bottle shattering is $1/2$ in $(M', u_{1,2})$ and 1 in both $(M', u_{1,0})$ and $(M', u_{1,1})$. Thus, the degree of blame of Suzy's throwing a rock for the bottle shattering is $5/6$.

3. Causes and explanations in first-order reasoning about actions

The work [9] presents a combination of the structural-model approach with first-order reasoning about actions in Poole's independent choice logic (ICL) [32,33]. It shows how the ICL can be extended by structure-based causes and explanations, and thus how structure-based concepts can be made available in first-order reasoning about actions. From another perspective, it also shows how first-order modeling capabilities and explicit actions can be added to the structural-model approach.

From a technical point of view, this combination is based on a mapping of first-order theories in the ICL to binary causal models via some grounding step. The generated causal models have a subset of the Herbrand base as set of endogenous variables, and thus they generally have a quite large number of variables. Hence, for this combination of the structural-model approach with first-order reasoning about actions in the ICL it is important to have efficient techniques for deciding and computing structure-based causal concepts in large causal models. An important structural feature of the large causal models that are generated from first-order theories in the ICL is that they have a natural layering through the time line. Hence, they are a special kind of *layered causal graphs*, which we will describe in Section 8 below as one important class of causal models for which deciding and computing causes and explanations is tractable under suitable restrictions.

Roughly, ICL-theories are defined as follows. A *choice space* C is a set of pairwise disjoint and nonempty subsets of the Herbrand base, called the *alternatives* of C. The elements of the alternatives of C are called the *atomic choices* of C. A *total choice* of C is a set of atomic choices B such that $|B \cap A| = 1$ for all alternatives A of C. An *independent choice logic theory* (or *ICL-theory*) $T = (C, L)$ consists of a choice space C and an acyclic logic program L such that no atomic choice in C coincides with the head of any clause in the grounding of L. Semantically, every total choice of C along with the acyclic logic program L produces a first-order model [9]. Hence, $T = (C, L)$ encodes the set of all such models. Every total choice and thus every first-order model is often also associated with a probability value.

It is not difficult to see that there is a natural relationship between binary structure-based causal models $M = (U, V, F)$ and ICL-theories $T = (C, L)$: (i) The exogenous variables in U along with their domains correspond to the alternatives of C along with their atomic choices, (ii) the endogenous variables in V along with their binary domains correspond to the ground atoms of T that do not act as atomic choices, along with their binary truth values, (iii) the functions in F correspond to collections of clauses with the same head in the grounding of L, and (iv) a probability function on the contexts in $D(U)$ corresponds to a probability function on the atomic choices of C. This natural relationship nicely supports the definition of structure-based causes and explanations in the ICL. The following example illustrates ICL-theories and structure-based causes in ICL-theories.

Example 3.1 *(mobile robot).* Consider a mobile robot, which can navigate in an environment and pick up objects. We assume the constants r_1 (robot), o_1 and o_2 (two objects), p_1 and p_2 (two positions), and $0, 1, \ldots, h$ (time points within

a *horizon* $h \geqslant 0$). The domain is described by the fluents $carrying(O, T)$ and $at(X, Pos, T)$, where $O \in \{o_1, o_2\}$, $T \in \{0, 1, \ldots, h\}$, $X \in \{r_1, o_1, o_2\}$, and $Pos \in \{p_1, p_2\}$, which encode that the robot r_1 is carrying the object O at time T (where we assume that at any time T the robot can hold at most one object), and that the robot or object X is at the position Pos at time T, respectively. The robot is endowed with the actions $moveTo(Pos)$, $pickUp(O)$, and $putDown(O)$, where $Pos \in \{p_1, p_2\}$ and $O \in \{o_1, o_2\}$, which represent the actions "move to the position P", "pick up the object O", and "put down the object O", respectively. The action $pickUp(O)$ is stochastic: It is not reliable, and thus can fail. Furthermore, we have the predicates $do(A, T)$, which represents the execution of an action A at time T, and $fa(A, T)$ (resp., $su(A, T)$), which represents the failure (resp., success) of an action A executed at time T. An ICL-theory (C, L) is then given as follows. The choice space C encodes that picking up an object $o_i \in \{o_1, o_2\}$ at time $t \in \{0, 1, \ldots, h\}$ may fail ($fa(pickUp(o_i), t)$) or succeed ($su(pickUp(o_i), t)$):

$$C = \big\{\{fa(pickUp(o_i), t), su(pickUp(o_i), t)\} \mid i \in \{1, 2\}, t \in \{0, 1, \ldots, h\}\big\}.$$

The acyclic logic program L consists of the clauses below, which encode the following knowledge:

- The robot is carrying the object O at time $T + 1$, if either (i) the robot and the object O were both at Pos at time T, and the robot was not carrying any object and successfully picking up the object O at time T, or (ii) the robot was carrying the object O and not putting it down at time T.

  (1) $carrying(O, T + 1) \Leftarrow (\neg carrying(o_1, T) \wedge \neg carrying(o_2, T) \wedge at(r_1, Pos, T) \wedge at(O, Pos, T) \wedge do(pickUp(O), T) \wedge su(pickUp(O), T)) \vee (carrying(O, T) \wedge \neg do(putDown(O), T))$.

- The robot is at Pos at time $T + 1$, if either (i) it moved to Pos at time T, or (ii) it was at Pos and did not move away at time T.

  (2) $at(r_1, Pos, T + 1) \Leftarrow do(moveTo(Pos), T) \vee (at(r_1, Pos, T) \wedge \neg do(moveTo(Pos'), T) \wedge Pos \neq Pos')$.

- The object O is at Pos at time $T + 1$, if either (i) the object was at Pos and not carried by the robot at time T, or (ii) the robot was carrying the object O and moved to Pos at time T, or (iii) the object was at Pos and carried by the robot, who did not move away at time T.

  (3) $at(O, Pos, T + 1) \Leftarrow (\neg carrying(O, T) \wedge at(O, Pos, T)) \vee (carrying(O, T) \wedge do(moveTo(Pos), T)) \vee (carrying(O, T) \wedge at(O, Pos, T) \wedge \neg do(moveTo(Pos'), T) \wedge Pos \neq Pos')$.

- The object o_1 is at the position p_2 at time 0.

  (4) $at(o_1, p_2, 0) \Leftarrow \top$.

- The robot is at the position p_2 at time 0.

  (5) $at(r_1, p_2, 0) \Leftarrow \top$.

Consider the horizon $h = 3$ and suppose that picking up an object succeeds at every time $t \in \{0, 1, 2, 3\}$, which is encoded by the total choice

$$B = \big\{su(pickUp(o_i), t) \mid i \in \{1, 2\}, t \in \{0, 1, 2, 3\}\big\}.$$

Suppose that the robot executes a pick up of o_1 at time 0, a move to p_1 at time 1, and a pick up of o_2 at time 2, which is represented by the additional facts

$$E = \big\{do(pickUp(o_1), 0), do(moveTo(p_1), 1), do(pickUp(o_2), 2)\big\}.$$

The structural-model approach now allows to give a semantics to causal statements in the ICL such as e.g. "the object o_1 being at position p_2 at time 0 is an actual cause of the robot not carrying the object o_2 at time 3 under the above B in $T \cup E$". Intuitively, the robot and the object o_1 are both at position p_2 at time 0. Hence, picking up o_1 succeeds at time 0, the robot moves with o_1 to position p_1 at time 1, there its picking up o_2 fails at time 2, and this

is why the robot is not carrying o_2 at time 3. However, if o_1 was not in position p_2 at time 0, and o_2 was always at position p_1, then the robot would hold no object at time 2, and its picking up o_2 at time 2 would succeed, and thus the robot would then be carrying o_2 at time 3.

Observe that the grounding step produces a causal model that has, even in this simple example, more than 90 variables (for a horizon $h \geqslant 0$, we have $24 \cdot (h+1)$ variables), which largely increases if we have more than only two positions and two objects different from the robot. Furthermore, the causal graph of this causal model is naturally layered through the time line (and thus has the structure of a layered causal graph as described in Section 8 below).

4. Problem statements

We concentrate on the following important computational problems for causes, explanations, responsibility, and blame in the structural-model approach, which comprise both decision problems and problems with concrete output.

4.1. Causes

WEAK/ACTUAL CAUSE: Given $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $u \in D(U)$, and an event ϕ, decide if $X = x$ is a weak (resp., an actual) cause of ϕ under u.

WEAK/ACTUAL CAUSE COMPUTATION: Given $M = (U, V, F)$, $X \subseteq V$, $u \in D(U)$, and an event ϕ, compute the set of all $X' = x'$ such that (i) $X' \subseteq X$ and $x' \in D(X')$, and (ii) $X' = x'$ is a weak (resp., an actual) cause of ϕ under u.

4.2. Notions of explanations

EXPLANATION: Given $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, an event ϕ, and a set of contexts $\mathcal{C} \subseteq D(U)$, decide whether $X = x$ is an explanation of ϕ relative to $\mathcal{C}$.

EXPLANATION COMPUTATION: Given $M = (U, V, F)$, $X \subseteq V$, an event ϕ, and a set of contexts $\mathcal{C} \subseteq D(U)$, compute the set of all $X' = x'$ such that (i) $X' \subseteq X$ and $x' \in D(X')$, and (ii) $X' = x'$ is an explanation of ϕ relative to $\mathcal{C}$.

α-PARTIAL EXPLANATION: Given $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, an event ϕ, a set of contexts $\mathcal{C} \subseteq D(U)$ such that $\phi(u)$ for all $u \in \mathcal{C}$, a probability function P on $\mathcal{C}$, and $\alpha \geqslant 0$, decide if $X = x$ is an α-partial explanation of ϕ relative to $(\mathcal{C}, P)$.

α-PARTIAL EXPLANATION COMPUTATION: Given $M = (U, V, F)$, $X \subseteq V$, an event ϕ, a set of contexts $\mathcal{C} \subseteq D(U)$ with $\phi(u)$ for all $u \in \mathcal{C}$, a probability function P on $\mathcal{C}$, and $\alpha \geqslant 0$, compute the set of all $X' = x'$ such that (i) $X' \subseteq X$ and $x' \in D(X')$, and (ii) $X' = x'$ is an α-partial explanation of ϕ relative to $(\mathcal{C}, P)$.

PARTIAL EXPLANATION: Given $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, an event ϕ, a set of contexts $\mathcal{C} \subseteq D(U)$ such that $\phi(u)$ for all $u \in \mathcal{C}$, and a probability function P on $\mathcal{C}$, decide whether $X = x$ is a partial explanation of ϕ relative to $(\mathcal{C}, P)$.

PARTIAL EXPLANATION COMPUTATION: Given $M = (U, V, F)$, $X \subseteq V$, an event ϕ, a set of contexts $\mathcal{C} \subseteq D(U)$ such that $\phi(u)$ for all $u \in \mathcal{C}$, and a probability function P on $\mathcal{C}$, compute the set of all $X' = x'$ such that (i) $X' \subseteq X$ and $x' \in D(X')$, and (ii) $X' = x'$ is a partial explanation of ϕ relative to $(\mathcal{C}, P)$.

EXPLANATORY POWER: Given $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, an event ϕ, a set of contexts $\mathcal{C} \subseteq D(U)$, and a probability function P on $\mathcal{C}$, where (i) $\phi(u)$ for all $u \in \mathcal{C}$, and (i) $X = x$ is a partial explanation of ϕ relative to $(\mathcal{C}, P)$, compute the explanatory power of $X = x$ for ϕ relative to $(\mathcal{C}, P)$.

4.3. Responsibility and blame

RESPONSIBILITY: Given $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $u \in D(U)$, and an event ϕ, compute the degree of responsibility of $X = x$ for ϕ in (M, u).

BLAME: Given an epistemic state $\mathcal{E}$, a set of endogenous variables X , $x \in D(X)$, and an event ϕ, compute the degree of blame of setting X to x for ϕ relative to $\mathcal{E}$.

4.4. Previous results

Several complexity results for the above problems have been established (considering the notion of weak cause as defined in [15,16]). In particular, as shown in [6], the decision problems WEAK CAUSE and ACTUAL CAUSE are both Σ_2^{P}-complete in the general case, and NP-complete in the case of binary variables. Furthermore, as shown in [7], the decision problems EXPLANATION and PARTIAL/α-PARTIAL EXPLANATION and the optimization problem EXPLANATORY POWER are complete for $\mathrm{D}_2^{\mathrm{P}}$, $\mathrm{P}_{\parallel}^{\Sigma_2^{\mathrm{P}}}$, and $\mathrm{FP}_{\parallel}^{\Sigma_2^{\mathrm{P}}}$, respectively, in the general case, and complete for D^{P}, $\mathrm{P}_{\parallel}^{\mathrm{NP}}$ and $\mathrm{FP}_{\parallel}^{\mathrm{NP}}$, respectively, in the binary case. Here $\mathrm{D}_2^{\mathrm{P}}$ (resp., D^{P}) is the "logical conjunction" of Σ_2^{P} and Π_2^{P} (resp., NP and co-NP), and $\mathrm{P}_{\parallel}^{C}$ (resp., $\mathrm{FP}_{\parallel}^{C}$) is the class of decision problems solvable (resp., functions computable) in polynomial time with access to one round of parallel queries to an oracle in C. Finally, Chockler and Halpern [3] and Chockler, Halpern, and Kupferman [4] have shown that the optimization problems RESPONSIBILITY and BLAME are complete for the classes $\mathrm{FP}^{\Sigma_2^{\mathrm{P}}[\log n]}$ and $\mathrm{FP}_{\parallel}^{\Sigma_2^{\mathrm{P}}}$, respectively, in the general case, and complete for $\mathrm{FP}^{\mathrm{NP}[\log n]}$ and $\mathrm{FP}_{\parallel}^{\mathrm{NP}}$, respectively, in the binary case. The class $\mathrm{FP}^{C[\log n]}$ contains the functions computable in polynomial time with $\mathrm{O}(\log n)$ many calls to an oracle in C, where n is the size of the problem input.

To our knowledge, there exist no complexity results for the optimization problems WEAK/ACTUAL CAUSE COMPUTATION, EXPLANATION COMPUTATION, and α-PARTIAL/PARTIAL EXPLANATION COMPUTATION so far. But there are complexity results on decision variants of two of the latter problems, which are called EXPLANATION EXISTENCE and α-PARTIAL EXPLANATION EXISTENCE, respectively. They are the decision problems of deciding whether an explanation and an α-partial explanation, respectively, over certain variables exists, which are complete for Σ_3^{P} (resp., Σ_2^{P}) in the general (resp., binary) case; see [7].

To our knowledge, there are no explicit tractability results for the above problems related to causes and explanations so far. As for responsibility and blame, Chockler, Halpern, and Kupferman [4] have shown that computing the degree of responsibility in read-once Boolean formulas can be done in linear time.

5. Irrelevant variables

In this section, we describe how an instance of deciding weak cause can be reduced with polynomial overhead to an equivalent instance in which the (potential) weak cause and the causal model may contain fewer variables. That is, such reductions identify and remove irrelevant variables in weak causes and also in causal models. This can be regarded as an important preliminary step in the computation of weak and actual causes, which seems to be indispensable in efficient implementations.

We first describe a reduction from [7] and a generalization thereof in which irrelevant variables in weak causes $X = x$ of an event ϕ are characterized and removed. We then generalize these two reductions to two new reductions that identify and remove irrelevant variables in weak causes $X = x$ of ϕ and also in causal models M, producing the *reduced* and the *strongly reduced causal model* of M w.r.t. $X = x$ and an event ϕ. Both new reductions also generalize a reduction due to Hopkins [21] for events of the form $X = x$ and $\phi = Y = y$, where X and Y are singletons. The reduced causal model of M w.r.t. $X = x$ and ϕ is in general larger than its strong reduct w.r.t. $X = x$ and ϕ. But the former allows for deciding whether $X' = x'$ is a weak cause of ϕ, for the large class of all $X' \subseteq X$, while the latter generally allows only for deciding whether $X = x$ is a weak cause of ϕ.

In the rest of this section, to illustrate the removal of variables in (potential) weak causes and causal models, we use what is shown in Fig. 2: (i) the causal graph $G_V(M)$ of a causal model $M = (U, V, F)$, (ii) the set of variables $X \subseteq V$ of a (potential) weak cause $X = x$, and (iii) the set of variables $V(\phi)$ in an event ϕ.

5.1. Reducing weak causes

The following result (essentially proved in [7]) shows that deciding whether $X = x$ is a weak cause of ϕ under u is reducible to deciding whether $X' = x|X'$ is a weak cause of ϕ under u, where X' is the set of all $X_i \in X$ that are either in ϕ or ancestors of variables in ϕ. That is, in deciding whether $X = x$ is a weak cause of ϕ under u, we can safely ignore all variables in $X = x$ not connected to any variable in ϕ.

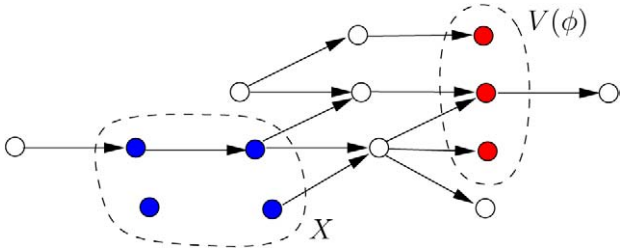

Fig. 2. Causal graph $G_V(M)$ along with X and $V(\phi)$.

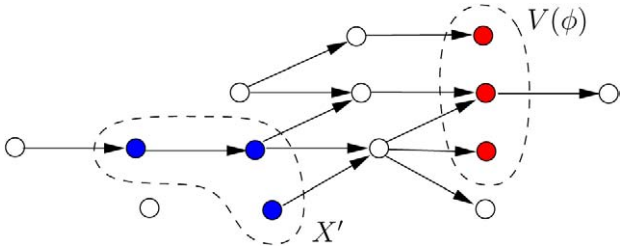

Fig. 3. Causal graph $G_V(M)$ along with X' and $V(\phi)$.

Theorem 5.1. (Essentially [7]) *Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$ and $x \in D(X)$, let ϕ be an event, and let $u \in D(U)$. Let X' be the set of all variables in X from which a (directed) path exists in $G(M)$ to a variable in ϕ, and let $x' = x|X'$. Then, $X = x$ is a weak cause of ϕ under u iff* (i) $(X \setminus X')(u) = x|(X \setminus X')$ *and* (ii) $X' = x'$ *is a weak cause of ϕ under u.*

Example 5.1. Fig. 3 shows X' for a causal model $M = (U, V, F)$ and an event ϕ such that the causal graph $G_V(M)$ and the sets X and $V(\phi)$ are as in Fig. 2.

The next theorem formulates the more general result that deciding whether $X = x$ is a weak cause of ϕ under u is reducible to deciding whether $X' = x|X'$ is a weak cause of ϕ under u, where X' is the set of all variables in X that occur in ϕ or that are ancestors of variables in ϕ not "blocked" by other variables in X. That is, in deciding whether $X = x$ is a weak cause of ϕ under u, we can even ignore every variable in $X = x$ not connected via variables in $V \setminus X$ to any variable in ϕ.

Theorem 5.2. *Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$ and $x \in D(X)$, let ϕ be an event, and let $u \in D(U)$. Let X' be the set of all variables $X_i \in X$ from which there exists a path in $G(M)$ to a variable in ϕ that contains no $X_j \in X \setminus \{X_i\}$, and let $x' = x|X'$. Then, $X = x$ is a weak cause of ϕ under u iff* (i) $(X \setminus X')(u) = x|(X \setminus X')$ *and* (ii) $X' = x'$ *is a weak cause of ϕ under u.*

Example 5.2. Fig. 4 shows X' for a causal model $M = (U, V, F)$ and an event ϕ such that the causal graph $G_V(M)$ and the sets X and $V(\phi)$ are as in Fig. 2.

The next result shows that computing the set of all variables in a weak cause that are not irrelevant according to Theorems 5.1 and 5.2 can be done in linear time.

Proposition 5.3. *Given a causal model $M = (U, V, F)$, $X \subseteq V$, and an event ϕ,*

(a) *computing the set X' of all variables $X_i \in X$ from which a path exists to a variable in ϕ can be done in time* $O(\|M\| + \|\phi\|)$;
(b) *computing the set X' of all variables $X_i \in X$ from which a path exists to a variable in ϕ that contains no $X_j \in X \setminus \{X_i\}$ can be done in time* $O(\|M\| + \|\phi\|)$.

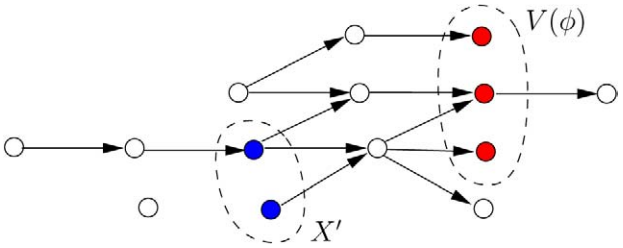

Fig. 4. Causal graph $G_V(M)$ along with X' and $V(\phi)$.

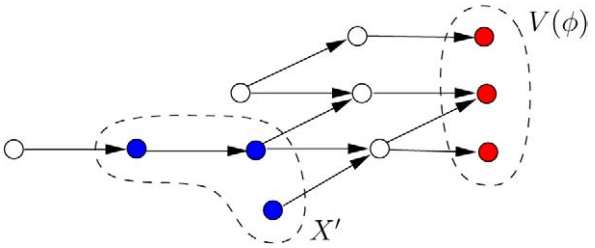

Fig. 5. Causal graph $G_V(M^{\phi}_X)$ along with $X' = X \cap R^{\phi}_X(M)$ and $V(\phi)$.

5.2. Reducing weak causes and causal models

We now generalize the reduction described in Theorem 5.1 to a reduction which not only removes irrelevant variables from causes, but also removes irrelevant variables in causal models. In the sequel, let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$, $x \in D(X)$, and $u \in D(U)$, and let ϕ be an event. We first define irrelevant variables w.r.t. $X = x$ and ϕ, and then the reduced causal model w.r.t. $X = x$ and ϕ, which does not contain these irrelevant variables anymore.

The set of *relevant* endogenous variables of $M = (U, V, F)$ w.r.t. $X = x$ and ϕ, denoted $R^{\phi}_{X=x}(M)$, is the set of all variables $A \in V$ from which there exists a directed path in $G(M)$ to a variable in ϕ. A variable $A \in V$ is *irrelevant* w.r.t. $X = x$ and ϕ iff it is not relevant w.r.t. $X = x$ and ϕ. Note that it does *not* necessarily hold that $X \subseteq R^{\phi}_{X=x}(M)$. The *reduced causal model* of $M = (U, V, F)$, where $F = \{F_A \mid A \in V\}$, w.r.t. $X = x$ and ϕ, denoted $M^{\phi}_{X=x}$, is the causal model $M' = (U, V', F')$, where $V' = R^{\phi}_{X=x}(M)$ and $F' = \{F'_A \mid A \in V'\}$ with $F'_A = F_A$ for all $A \in V'$. We often use $R^{\phi}_X(M)$, $R^Y_X(M)$, M^{ϕ}_X, and M^Y_X to abbreviate $R^{\phi}_{X=x}(M)$, $R^{Y=y}_{X=x}(M)$, $M^{\phi}_{X=x}$, and $M^{Y=y}_{X=x}$, respectively.

Example 5.3. Fig. 5 shows the causal graph $G_V(M^{\phi}_X)$ along with the set of variables $X' = X \cap R^{\phi}_X(M)$ for a causal model $M = (U, V, F)$ and an event ϕ such that the causal graph $G_V(M)$ and the sets X and $V(\phi)$ are as in Fig. 2.

The following result shows that a variable in $X = x$ is irrelevant w.r.t. $X = x$ and ϕ iff it is not connected to a variable in ϕ. Hence, we are heading towards a generalization of the reduction in Theorem 5.1.

Proposition 5.4. *Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$, $x \in D(X)$, and let ϕ be an event. Then, $X \cap R^{\phi}_X(M)$ is the set of all variables $B \in X$ from which there exists a directed path in $G(M)$ to a variable in ϕ.*

The next result shows that deciding whether $X = x$ is a weak cause of ϕ under u in M can be reduced to deciding whether $X' = x'$ is a weak cause of ϕ under u in M^{ϕ}_X, where $X' = X \cap R^{\phi}_X(M)$ and $x' = x|X'$. It generalizes Theorem 5.1. Note that this result and also Theorems 5.7 and 5.10 below do not carry over to responsibility.

Theorem 5.5. *Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$, $x \in D(X)$, and $u \in D(U)$, and let ϕ be an event. Let $X' = X \cap R^{\phi}_X(M)$ and $x' = x|X'$. Then, $X = x$ is a weak cause of ϕ under u in M iff* (i) *$(X \setminus X')(u) = x|(X \setminus X')$ in M, and* (ii) *$X' = x'$ is a weak cause of ϕ under u in M^{ϕ}_X.*

Our next result shows that the reduction of a causal model is independent from X. Informally, the reduced causal model of M w.r.t. $X = x$ and ϕ coincides with the reduced causal model of M w.r.t. $X' = x'$ and ϕ.

Proposition 5.6. *Let $M = (U, V, F)$ be a causal model. Let $X, X' \subseteq V$, $x \in D(X)$, and $x' \in D(X')$, and let ϕ be an event. Then, M^{ϕ}_{X} coincides with $M^{\phi}_{X'}$.*

We are now ready to formulate the main result of this section. The following theorem shows that deciding whether $X' = x'$, where $X' \subseteq X$, is a weak cause of ϕ under u in M can be reduced to deciding whether its restriction to $R^{\phi}_{X}(M)$ is a weak cause of ϕ under u in M^{ϕ}_{X}. It is a generalization of Theorems 5.1 and 5.5, which follows from Theorem 5.5 and Proposition 5.6.

Theorem 5.7. *Let $M = (U, V, F)$ be a causal model. Let $X' \subseteq X \subseteq V$, $x' \in D(X')$, $x \in D(X)$, and $u \in D(U)$, and let ϕ be an event. Let $X'' = X' \cap R^{\phi}_{X}(M)$ and $x'' = x'|X''$. Then, $X' = x'$ is a weak cause of ϕ under u in M iff* (i) *$(X' \setminus X'')(u) = x'|(X' \setminus X'')$ in M, and* (ii) *$X'' = x''$ is a weak cause of ϕ under u in M^{ϕ}_{X}.*

The following result shows that the reduced causal model and the restriction of its causal graph to the set of endogenous variables can be computed in linear time.

Proposition 5.8. *Given a causal model $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, and an event ϕ, the directed graph $G_V(M^{\phi}_{X})$ and the causal model M^{ϕ}_{X} can be computed in time* $O(\|M\| + \|\phi\|)$.

5.3. Strongly reducing weak causes and causal models

In the sequel, let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$, $x \in D(X)$, and $u \in D(U)$, and let ϕ be an event. The reduced causal model w.r.t. $X = x$ and ϕ, which generalizes the idea behind Theorem 5.1, still contains some superfluous variables for deciding whether $X = x$ is a weak causes of ϕ under u in M. We now define the strongly reduced causal model w.r.t. $X = x$ and ϕ, which generalizes the idea behind Theorem 5.2, where these superfluous variables are removed. We first define strongly relevant variables w.r.t. $X = x$ and ϕ, and then the strongly reduced causal model w.r.t. $X = x$ and ϕ, which contains only such variables.

The set of *strongly relevant* endogenous variables of $M = (U, V, F)$ w.r.t. $X = x$ and ϕ, denoted $\widehat{R}^{\phi}_{X=x}(M)$, is the set of all variables $A \in V$ from which there exists a directed path in $G(M)$ to a variable in ϕ that contains no $X_j \in X \setminus \{A\}$. Observe that $\widehat{R}^{\phi}_{X=x}(M) \subseteq R^{\phi}_{X=x}(M)$. The *strongly reduced causal model* of $M = (U, V, F)$, where $F = \{F_A \mid A \in V\}$, w.r.t. $X = x$ and ϕ, denoted $\widehat{M}^{\phi}_{X=x}$, is the causal model $M' = (U, V', F')$, where $V' = \widehat{R}^{\phi}_{X=x}(M)$ and $F' = \{F'_A \mid A \in V'\}$ with $F'_A = F^{\star}_A$ for all $A \in V' \cap X$ and $F'_A = F_A$ for all $A \in V' \setminus X$. Here, $F^{\star}_A$ assigns $A_M(u_A)$ to A for every value $u_A \in D(U_A)$ of the set U_A of all ancestors $B \in U$ of A in $G(M)$. We often use $\widehat{R}^{\phi}_{X}(M)$, $\widehat{R}^{Y}_{X}(M)$, $\widehat{M}^{\phi}_{X}$, and $\widehat{M}^{Y}_{X}$ to abbreviate $\widehat{R}^{\phi}_{X=x}(M)$, $\widehat{R}^{Y=y}_{X=x}(M)$, $\widehat{M}^{\phi}_{X=x}$, and $\widehat{M}^{Y=y}_{X=x}$, respectively.

Example 5.4. Fig. 6 shows the causal graph $G_V(\widehat{M}^{\phi}_{X})$ along with the set of variables $X' = X \cap \widehat{R}^{\phi}_{X}(M)$ for a causal model $M = (U, V, F)$ and an event ϕ such that the causal graph $G_V(M)$ and the sets X and $V(\phi)$ are as in Fig. 2.

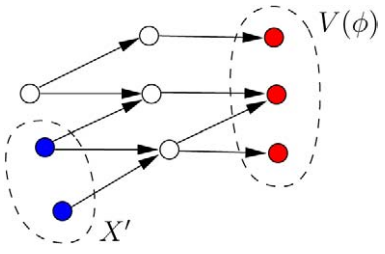

Fig. 6. Causal graph $G_V(\widehat{M}^{\phi}_{X})$ along with $X' = X \cap \widehat{R}^{\phi}_{X}(M)$ and $V(\phi)$.

The following result shows that a variable in $X = x$ is strongly relevant w.r.t. $X = x$ and ϕ iff it is connected in $G(M)$ via variables in $V \setminus X$ to a variable in ϕ. Thus, we are currently elaborating a generalization of Theorem 5.2.

Proposition 5.9. *Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$, $x \in D(X)$, and let ϕ be an event. Then, $X \cap \widehat{R}^{\phi}_{X}(M)$ is the set of all $X_i \in X$ from which there exists a directed path in $G(M)$ to a variable in ϕ that contains no $X_j \in X \setminus \{X_i\}$.*

It is easy to verify that both Proposition 5.6 and Theorem 5.7 do not carry over to strongly reduced causal models. Informally, if $X' \subseteq X$, then $\widehat{M}^{\phi}_{X'}$ may contain variables that connect some $X_i \in V$ to a variable in ϕ *via variables in* $V \setminus X'$, but that do not connect $X_i \in V$ to a variable in ϕ *via variables in* $V \setminus X \subseteq V \setminus X'$, since some variable from $X \setminus X'$ is needed, and are thus not contained in $\widehat{M}^{\phi}_{X}$. For example, if the causal graph $G_V(M)$ and the sets X and $V(\phi)$ are as in Fig. 2, and X' consists of the variable in X that is shown upper left in Fig. 2, then this variable does not occur among the variables of the strongly reduced causal model $\widehat{M}^{\phi}_{X}$, since it is pruned away (cf. also Fig. 6).

However, the weaker result in Theorem 5.5 also holds for strongly reduced causal models. That is, deciding whether $X = x$ is a weak cause of ϕ under u in M can be reduced to deciding whether its restriction to the strongly relevant variables is a weak cause of ϕ under u in $\widehat{M}^{\phi}_{X}$. This result generalizes Theorem 5.2.

Theorem 5.10. *Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$, $x \in D(X)$, and $u \in D(U)$, and let ϕ be an event. Let $X' = X \cap \widehat{R}^{\phi}_{X}(M)$ and $x' = x|X'$. Then, $X = x$ is a weak cause of ϕ under u in M iff* (i) *$(X \setminus X')(u) = x|(X \setminus X')$ in M, and* (ii) *$X' = x'$ is a weak cause of ϕ under u in $\widehat{M}^{\phi}_{X}$.*

The following result shows that the strongly reduced causal model and the restriction of its causal graph to the set of all endogenous variables can be computed in polynomial and linear time, respectively. Here, for any set S, we denote by $|S|$ its cardinality.

Proposition 5.11. *Given a causal model $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, and an event ϕ, the directed graph $G_V(\widehat{M}^{\phi}_{X})$ (resp., the causal model $\widehat{M}^{\phi}_{X}$) is computable in time $\mathrm{O}(\|M\| + \|\phi\|)$ (resp., $\mathrm{O}(|V|\|M\| + \|\phi\|)$).*

6. Causal trees

In this section, we describe our first class of tractable cases of causes and explanations. We show that deciding whether an atom $X = x$ is a weak cause of a primitive event $Y = y$ under a context u in a domain-bounded causal model $M = (U, V, F)$ is tractable, if the strongly reduced causal model $G_V(\widehat{M}^{Y}_{X})$ is a bounded directed tree with root Y, which informally consists of a directed path from X to Y, along with a number of parents for each variable in the path after X bounded by a global constant (see Fig. 7). Under the same conditions, deciding whether $X = x$ is an actual cause, deciding whether $X = x$ is an explanation relative to a set of contexts $\mathcal{C}$, and deciding whether $X = x$ is a partial explanation or an α-partial explanation as well as computing its explanatory power relative to $(\mathcal{C}, P)$ are all tractable.

More precisely, we say that a directed graph $G = (V, E)$, given two nodes $X, Y \in V$, is a *directed tree with root Y*, if it consists of a unique directed path $X \hat{=} P^k \to P^{k-1} \to \cdots \to P^0 \hat{=} Y$ from X to Y, and sets W^i of (unconnected) parents $A \neq P^i$ for all P^{i-1} such that $i \in \{1, \ldots, k\}$. Moreover, G is *bounded*, if $|W_i| \leqslant l$ for each $i \in \{1, \ldots, k\}$, i.e., P_{i-1} has fan-in of variables from V at most $l + 1$, where l is some global constant. If $G = G_V(M)$ for some causal model $M = (U, V, F)$ and $X, Y \in V$, then M is a (*bounded*) *causal tree* with respect to X and Y.

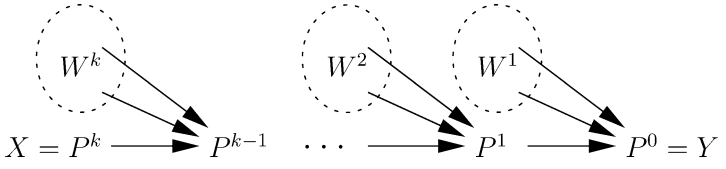

Fig. 7. Path from X to Y in a causal tree.

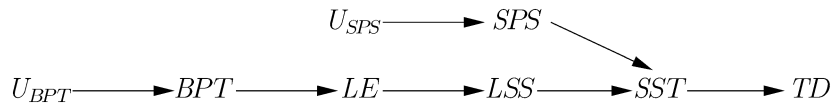

Fig. 8. Causal graph $G(M)$.

Example 6.1. An example of a causal tree is the following binary causal model $M = (U, V, F)$ presented in [16] in a discussion of the double prevention problem, where $U = \{U_{BPT}, U_{SPS}\}$ with $D(A) = \{0, 1\}$ for all $A \in U$, $V = \{BPT, LE, LSS, SPS, SST, TD\}$ with $D(A) = \{0, 1\}$ for all $A \in V$. In a World War III scenario, Suzy is piloting a bomber on a mission to blow up an enemy target, and Billy is piloting a fighter as her lone escort. Along comes an enemy plane, piloted by Lucifer. Sharp-eyed Billy spots Lucifer, zooms in, pulls the trigger, and Lucifer's plane goes down in flames. Suzy's mission is undisturbed, and the bombing takes place as planned. The question is whether Billy deserves some credit for the success of the mission. Here, *BPT* means that Billy pulls the trigger, *LE* that Lucifer eludes Billy, *LSS* that Lucifer shoots Suzy, *SPS* that Suzy plans to shoot the target, *SST* that Suzy shoots the target, and *TD* that the target is destroyed.

The set $F = \{F_A \mid A \in V\}$ consists of the functions $F_{BPT} = U_{BPT}$, $F_{SPS} = U_{SPS}$, $F_{LE} = 1 - BPT$, $F_{LSS} = LE$, $F_{SST} = 1$ iff $LSS = 0$ and $SPS = 1$, and $F_{TD} = SST$. The causal graph $G(M)$ is shown in Fig. 8. Let $X = BPT$ and $Y = TD$. Then, $G_V(M)$ is a directed tree with root Y, where the directed path from X to Y is $P^4 = BPT$, $P^3 = LE$, $P^2 = LSS$, $P^1 = SST$, $P^0 = TD$, $W^1 = W^3 = W^4 = \emptyset$ and $W^2 = SPS$.

As an important property, causal trees can be recognized very efficiently, namely in linear time. The same holds for causal models whose reduced variant with respect to X and Y is a causal tree.

Proposition 6.1. *Given a causal model $M = (U, V, F)$ and variables $X, Y \in V$, deciding whether M resp. $\widehat{M}_Y^X$ is a (unbounded or bounded) causal tree with respect to X and Y is feasible in $O(\|M\|)$ time.*

6.1. Characterizing weak causes

We first consider weak causes. In the sequel, let $M = (U, V, F)$ be a causal model, let $X, Y \in V$ such that M is a causal tree with respect to X and Y, and let $x \in D(X)$ and $y \in D(Y)$. We give a new characterization of $X = x$ being a weak cause of $Y = y$ under context $u \in D(U)$, which can be checked in polynomial time under some conditions. We need some preparative definitions. We define $R^0 = \{(D(Y) \setminus \{y\}, \{y\})\}$, and for every $i \in \{1, \ldots, k\}$, we define $\hat{p}^i = P^i(u)$ and R^i by:

$$\begin{aligned} R^i = \big\{(\boldsymbol{p}, \boldsymbol{q}) \mid \boldsymbol{p}, \boldsymbol{q} \subseteq D(P^i), \exists w \in D(W^i) \exists (\boldsymbol{p'}, \boldsymbol{q'}) \in R^{i-1}\ \forall p, q \in D(P^i)\colon \\ p \in \boldsymbol{p} \text{ iff } P^{i-1}_{pw}(u) \in \boldsymbol{p'}, \\ q \in \boldsymbol{q} \text{ iff } P^{i-1}_{qw'}(u), P^{i-1}_{\hat{p}^j w'}(u) \in \boldsymbol{q'} \text{ for all } W' \subseteq W^i, w' = w|W', \text{ and } j \in \{1, \ldots, \min(i, k-1)\}\big\}. \end{aligned}$$

Roughly, R^i is the set of all pairs $(\boldsymbol{p}, \boldsymbol{q})$, where $\boldsymbol{p}$ (resp., $\boldsymbol{q}$) is a set of possible values of P^i in **AC2**(a) (resp., (b)), for different values of W^i. Here, $P^0 \hat{=}\ Y$ must be set to a value different from y (resp., to the value y), and the possible values of each other P^i depend on the possible values of P^{i-1}. In summary, **AC2**(a) and (b) hold iff some $(\boldsymbol{p}, \boldsymbol{q}) \in R^k$ exists such that $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$. This result is formally expressed by the following theorem, which can be proved by induction on $i \in \{0, \ldots, k\}$.

Theorem 6.2. *Let $M = (U, V, F)$ be a causal model. Let $X, Y \in V$, $x \in D(X)$, $y \in D(Y)$, and $u \in D(U)$. Suppose M is a causal tree w.r.t. X and Y, and let R^k be defined as above. Then, $X = x$ is a weak cause of $Y = y$ under u in M iff (α) $X(u) = x$ and $Y(u) = y$ in M, and (β) some $(\boldsymbol{p}, \boldsymbol{q}) \in R^k$ exists with $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$.*

Example 6.2. Consider again the causal tree with respect to $X = BPT$ and $Y = TD$ from Example 6.1. Suppose we want to decide whether $BPT = 1$ is a weak cause of $TD = 1$ under a context $u_{1,1} \in D(U)$, where $u_{1,1}(U_{BPT}) = 1$ and $u_{1,1}(U_{SPS}) = 1$. Here, we obtain the relations $R^0 = \{(\{0\}, \{1\})\}$, $R^1 = \{(\{0\}, \{1\})\}$, $R^2 = \{(\{1\}, \{0\})\}$, $R^3 = \{(\{1\}, \{0\})\}$, and $R^4 = \{(\{0\}, \{1\})\}$. Notice then that (α) $BPT(u_{1,1})$ and $TD(u_{1,1})$ are both 1, and (β) $(\{0\}, \{1\}) \in R^4$, $\{0\} \neq \emptyset$, and $1 \in \{1\}$. By Theorem 6.2, it thus follows that $BPT = 1$ is a weak cause of $TD = 1$ under $u_{1,1}$.

6.2. Deciding weak and actual causes

The following theorem shows that deciding whether an atom $X = x$ is a weak cause of a primitive event $Y = y$ in domain-bounded M is tractable, when M is a bounded causal tree with respect to X and Y. This result follows from Theorem 6.2 and the recursive definition of R^i, which assures that R^k can be computed in polynomial time under the above boundedness assumptions. By Theorem 2.3, the same tractability result holds for actual causes, since for singletons X the notion of actual cause coincides with the notion of weak cause.

Theorem 6.3. *Given a domain-bounded causal model $M = (U, V, F)$, variables $X, Y \in V$ such that M is a bounded causal tree with respect to X and Y, and values $x \in D(X)$, $y \in D(Y)$, and $u \in D(U)$, deciding whether $X = x$ is a weak (resp., an actual) cause of $Y = y$ under u in M can be done in polynomial time.*

The next theorem shows that the same tractability result holds when instead of M just the reduced model $\widehat{M}_Y^X$ is required to be a bounded causal tree. The result follows from Theorem 5.10, Proposition 5.11, and Theorem 6.3.

Theorem 6.4. *Given a domain-bounded causal model $M = (U, V, F)$, variables $X, Y \in V$ such that $\widehat{M}_X^Y$ is a bounded causal tree with respect to X and Y, values $x \in D(X)$, $y \in D(Y)$, and $u \in D(U)$, deciding whether $X = x$ is a weak (resp., an actual) cause of $Y = y$ under u in M can be done in polynomial time.*

6.3. Deciding explanations and partial explanations

The following theorem shows that deciding whether $X = x$ is an explanation of $Y = y$ relative to $\mathcal{C}$ in M is tractable under the conditions of the previous subsection. This result follows from Theorem 6.4 and Proposition 2.2.

Theorem 6.5. *Given a domain-bounded causal model $M = (U, V, F)$, variables $X, Y \in V$ such that $\widehat{M}_X^Y$ is a bounded causal tree with respect to X and Y, values $x \in D(X)$ and $y \in D(Y)$, and a set of contexts $\mathcal{C} \subseteq D(U)$, deciding whether $X = x$ is an explanation of $Y = y$ relative to $\mathcal{C}$ in M can be done in polynomial time.*

Similarly, deciding whether $X = x$ is a partial or an α-partial explanation of $Y = y$ relative to $(\mathcal{C}, P)$ in M, as well as computing its explanatory power is tractable under the conditions of the previous subsection. This follows from Theorem 6.4 and Propositions 2.2 and 2.4.

Theorem 6.6. *Let $M = (U, V, F)$ be a domain-bounded causal model, let $X, Y \in V$ be such that $\widehat{M}_X^Y$ is a bounded causal tree with respect to X and Y, and let $x \in D(X)$ and $y \in D(Y)$. Let $\mathcal{C} \subseteq D(U)$ such that $Y(u) = y$ for all $u \in \mathcal{C}$, and let P be a probability function on $\mathcal{C}$. Then,*

(a) *deciding whether $X = x$ is a partial explanation of $Y = y$ relative to $(\mathcal{C}, P)$ in M can be done in polynomial time*;
(b) *deciding whether $X = x$ is an α-partial explanation of $Y = y$ relative to $(\mathcal{C}, P)$ in M, for some given $\alpha \geqslant 0$ can be done in polynomial time*;
(c) *given that $X = x$ is a partial explanation of $Y = y$ relative to $(\mathcal{C}, P)$ in M, the explanatory power of $X = x$ is computable in polynomial time.*

7. Decomposable causal graphs

In this section, we generalize the characterization of weak cause given in Section 6 to more general events and to more general causal graphs. We characterize relationships of the form "$X = x$ is a weak cause of ϕ under u in M", where (i) $X = x$ and ϕ are as in the original definition of weak cause, and thus not restricted to assignments to single variables anymore, and where (ii) $G_V(M)$ is decomposable into a chain of subgraphs (cf. Fig. 9, which is explained in more detail below), and thus not restricted to causal trees anymore. We then use this result to obtain more general tractability results for causes and explanations, and also new tractability results for responsibility and blame.

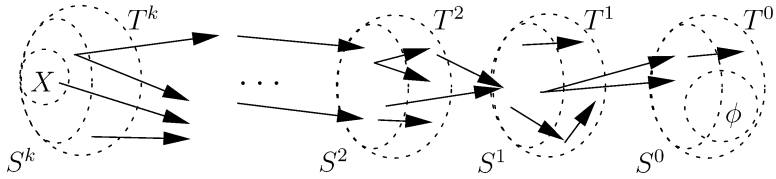

Fig. 9. Decomposable causal graph.

7.1. Characterizing weak causes

We first give a new characterization of weak cause. In the sequel, let $M = (U, V, F)$ be a causal model, let $X \subseteq V$, let $x \in D(X)$ and $u \in D(U)$, and let ϕ be an event.

Towards a characterization of "$X = x$ is a weak cause of ϕ under u in M", we define the notion of a decomposition of a causal graph as follows. A *decomposition* of $G_V(M)$ relative to $X = x$ (or simply X) and ϕ is a tuple $((T^0, S^0), \ldots, (T^k, S^k))$, $k \geqslant 0$, of pairs (T^i, S^i) such that the conditions **D1**–**D6** hold:

D1. $(T^0, \ldots, T^k)$ is an ordered partition of V.
D2. $T^0 \supseteq S^0, \ldots, T^k \supseteq S^k$.
D3. Every $A \in V$ occurring in ϕ belongs to T^0, and $S^k \supseteq X$.
D4. For every $i \in \{0, \ldots, k-1\}$, no two variables $A \in T^0 \cup \cdots \cup T^{i-1} \cup T^i \setminus S^i$ and $B \in T^{i+1} \cup \cdots \cup T^k$ are connected by an arrow in $G_V(M)$.
D5. For every $i \in \{1, \ldots, k\}$, every child of a variable from S^i in $G_V(M)$ belongs to $(T^i \setminus S^i) \cup S^{i-1}$. Every child of a variable from S^0 belongs to $(T^0 \setminus S^0)$.
D6. For every $i \in \{0, \ldots, k-1\}$, every parent of a variable in S^i in $G_V(M)$ belongs to T^{i+1}. There are no parents of any variable $A \in S^k$.

Intuitively, $G_V(M)$ is decomposable into a chain of edge-disjoint subgraphs $G^0, \ldots, G^k$ over some sets of variables $T^0, S^0 \cup T^1, S^1 \cup T^2, \ldots, S^{k-1} \cup T^k$, where $(T^0, \ldots, T^k)$ is an ordered partition of V, such that the sets T_i are connected to each other exactly through some sets $S^i \subseteq T^i$, $i \in \{0, \ldots, k-1\}$, where (i) every arrow that is incident to some $A \in S^i$, $i \in \{1, \ldots, k-1\}$, is either outgoing from A and belongs to G^i, or ingoing into A and belongs to G^{i+1}, and (ii) every variable in ϕ (resp., X) belongs to T^0 (resp., some $S^k \subseteq T^k$); see Fig. 9 for an illustration.

As easily seen, causal trees as in Section 6 are causal models with decomposable causal graphs. For the directed path $X \mathrel{\hat{=}} P^k \to P^{k-1} \to \cdots \to P^0 \mathrel{\hat{=}} Y$ from X to Y, and the sets W^i, $i \in \{1, \ldots, k\}$, we may define $\mathcal{D} = ((T^0, S^0), \ldots, (T^k, S^k))$ by $S^i = \{P^i\}$, $T^0 = \{P^0\}$, and $T^i = W^i \cup \{P^i\}$, for $i \in \{1, \ldots, k\}$; then, $\mathcal{D}$ is a decomposition of $G_V(M)$ relative to $X = x$ and $Y = y$.

The *width* of a decomposition $\mathcal{D} = ((T^0, S^0), \ldots, (T^k, S^k))$ of $G_V(M)$ relative to X and ϕ is the maximum of all $|T^i|$ such that $i \in \{0, \ldots, k\}$. We say that $\mathcal{D}$ is *width-bounded* iff the width of $\mathcal{D}$ is at most l for some global constant l.

Example 7.1. Fig. 10 shows a decomposition $\mathcal{D} = ((T^0, S^0), (T^1, S^1), (T^2, S^2))$ of a causal graph $G_V(M)$ relative to a set of variables $X \subseteq V$ and an event ϕ. The width of this decomposition $\mathcal{D}$ is given by 6.

We use such a decomposition $((T^0, S^0), \ldots, (T^k, S^k))$ of $G_V(M)$ to extend the relations R^i for causal trees from Section 6.1 to decomposable causal graphs. The new relations R^i now contain triples $(\boldsymbol{p}, \boldsymbol{q}, F)$, where $\boldsymbol{p}$ (resp., $\boldsymbol{q}$) specifies a set of possible values of "floating variables" $F \subseteq S^i$ in **AC2**(a) (resp., (b)). We define R^0 as follows:

$$
\begin{aligned}
R^0 = \big\{(\boldsymbol{p}, \boldsymbol{q}, F) \mid\ & F \subseteq S^0,\ \boldsymbol{p}, \boldsymbol{q} \subseteq D(F), \\
& \exists W \subseteq T^0,\ F = S^0 \setminus W, \\
& \exists w \in D(W)\ \forall p, q \in D(F): \\
& p \in \boldsymbol{p} \text{ iff } \neg\phi_{pw}(u), \\
& q \in \boldsymbol{q} \text{ iff } \phi_{[q\langle \hat{Z}(u)\rangle w']}(u) \text{ for all } W' \subseteq W,\ w' = w|W',\ \text{and } \hat{Z} \subseteq T^0 \setminus (S^k \cup W)\big\}.
\end{aligned}
$$

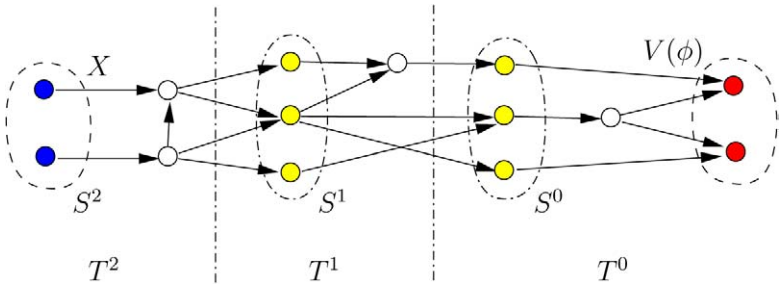

Fig. 10. Decomposition $((T^0, S^0), (T^1, S^1), (T^2, S^2))$ of $G_V(M)$ relative to X and ϕ.

For every $i \in \{1, \ldots, k\}$, we then define R^i as follows:

$$\begin{aligned} R^i = \big\{ (\boldsymbol{p}, \boldsymbol{q}, F) \mid\ & F \subseteq S^i,\ \boldsymbol{p}, \boldsymbol{q} \subseteq D(F), \\ & \exists W \subseteq T^i,\ F = S^i \setminus W, \\ & \exists w \in D(W)\ \exists (\boldsymbol{p'}, \boldsymbol{q'}, F') \in R^{i-1}\ \forall p, q \in D(F): \\ & p \in \boldsymbol{p} \text{ iff } {F'}_{pw}(u) \in \boldsymbol{p'}, \\ & q \in \boldsymbol{q} \text{ iff } {F'}_{[q\langle \hat{Z}(u)]w'}(u) \in \boldsymbol{q'} \text{ for all } W' \subseteq W,\ w' = w|W',\ \text{and } \hat{Z} \subseteq T^i \setminus (S^k \cup W) \big\}. \end{aligned}$$

Intuitively, rather than propagating sets of pairs $(\boldsymbol{p}, \boldsymbol{q})$ of possible values $\boldsymbol{p}$ and $\boldsymbol{q}$ of a single variable $P^i \in V$ in **AC2**(a) and (b), respectively, from $Y = P^0$ back to $X = P^k$ along a path $X \hat{=} P^k \to P^{k-1} \to \cdots \to P^0 \hat{=} Y$ as in Section 6.1, we now propagate triples $(\boldsymbol{p}, \boldsymbol{q}, F)$ consisting of some "floating variables" $F \subseteq S^i \subseteq V$, a set $\boldsymbol{p}$ of possible values of F in **AC2**(a), and a set $\boldsymbol{q}$ of possible values of F in **AC2**(b), from ϕ back to $X \subseteq V$ along the chain of subgraphs $G^0, \ldots, G^k$ over the sets of variables T^0, $S^0 \cup T^1$, $S^1 \cup T^2, \ldots,$ $S^{k-1} \cup T^k$. Here, R^0 contains all triples $(\boldsymbol{p}, \boldsymbol{q}, F)$ such that $F \subseteq S^0$, $\boldsymbol{p}, \boldsymbol{q} \subseteq D(F)$, $p \in \boldsymbol{p}$ iff $\neg \phi_{pw}(u)$, and $q \in \boldsymbol{q}$ iff $\phi_{[q\langle \hat{Z}(u)]w'}(u)$, for all possible $\hat{Z}$ and restrictions w' of some appropriate w. Moreover, the triples in R^i depend on the triples in R^{i-1}. In summary, it then follows that **AC2**(a) and (b) hold iff some $(\boldsymbol{p}, \boldsymbol{q}, X) \in R^k$ exists such that $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$.

Note that for a decomposition corresponding to a causal tree as discussed above, for each $(\boldsymbol{p}, \boldsymbol{q}, F)$ in R^0, it holds that $W = \emptyset$ and $F = \{P^0\}$. Furthermore, for each $(\boldsymbol{p}, \boldsymbol{q}, F)$ in R^i, where $i > 0$, we have $W = W^i$ and $F = \{P^i\}$. That is, the sets R^i defined for causal trees correspond to simplified versions of the sets R^i for a decomposed graph, where the redundant component F is removed from each triple.

This new characterization of weak cause, which is based on the above concept of a decomposition of $G_V(M)$ and the relations R^i, is expressed by the following theorem, which can be proved by induction on $i \in \{0, \ldots, k\}$.

Theorem 7.1. *Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$, let $x \in D(X)$ and $u \in D(U)$, and let ϕ be an event. Let $((T^0, S^0), \ldots, (T^k, S^k))$ be a decomposition of $G_V(M)$ relative to X and ϕ, and let R^k be defined as above. Then, $X = x$ is a weak cause of ϕ under u in M iff (α) $X(u) = x$ and $\phi(u)$ in M, and (β) some $(\boldsymbol{p}, \boldsymbol{q}, X) \in R^k$ exists such that $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$.*

This result provides a basis for deciding and computing weak and actual causes, and may in particular be fruitfully applied to reduced causal models from which irrelevant variables have been pruned. Often, reduced models have a simple decomposition: Every $\widehat{M}^{\phi}_X = (U, V', F')$ has the trivial decomposition $((V', X))$, and every $M^{\phi}_X = (U, V', F')$ such that no $A \in X$ is on a path from a different variable in X to a variable in ϕ also has the trivial decomposition $((V', X))$.

7.2. Deciding and computing weak and actual causes

Using the characterization of weak cause given in Section 7.1, we now provide new tractability results for deciding and computing weak and actual causes. The following theorem shows that deciding whether $X = x$ is a weak (resp., an actual) cause of ϕ under u in a domain-bounded M is tractable when $G_V(M)$ has a width-bounded decomposition

provided in the input. As for its proof, by Theorem 7.1, deciding whether $X = x$ is a weak cause of ϕ under u in M can be done by recursively computing the R^i's and then deciding whether (α) and (β) of Theorem 7.1 hold. All this can be done in polynomial time under the above boundedness assumptions. Since $G_V(M)$ has a width-bounded decomposition, $|X|$ is bounded by a constant, and thus the above tractability result also holds for actual causes.

Theorem 7.2. *Given a domain-bounded causal model $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $u \in D(U)$, an event ϕ, and a width-bounded decomposition $\mathcal{D}$ of $G_V(M)$ relative to X and ϕ, deciding whether $X = x$ is a weak (resp., an actual) cause of ϕ under u in M is possible in polynomial time.*

The next theorem shows that deciding weak (resp., actual) causes in domain-bounded causal models is also tractable, when $G_V(M^{\phi}_X)$ has a width-bounded decomposition provided in the input. This result essentially combines Theorems 5.7 and 7.2.

Theorem 7.3. *Given a domain-bounded causal model $M = (U, V, F)$, $X' \subseteq X \subseteq V$, $x' \in D(X')$, $u \in D(U)$, an event ϕ, and a width-bounded decomposition $\mathcal{D}$ of the graph $G_V(M^{\phi}_X)$ relative to $X' \cap R^{\phi}_X(M)$ and ϕ, deciding whether $X' = x'$ is a weak (resp., an actual) cause of ϕ under u in M is possible in polynomial time.*

A similar result also holds for strongly reduced causal models. It is expressed by the following theorem, which basically combines Theorems 5.10 and 7.2.

Theorem 7.4. *Given a domain-bounded causal model $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $u \in D(U)$, an event ϕ, and a width-bounded decomposition $\mathcal{D}$ of the graph $G_V(\widehat{M}^{\phi}_X)$ relative to $X \cap \widehat{R}^{\phi}_X(M)$ and ϕ, deciding whether $X = x$ is a weak (resp., an actual) cause of ϕ under u in M is possible in polynomial time.*

We finally focus on computing weak and actual causes. Th e following result shows that, given some $X \subseteq V$, computing all weak (resp., actual) causes $X' = x'$, where $X' \subseteq X$ and $x' \in D(X')$, of ϕ under u in domain-bounded M is tractable, when either (a) $G_V(M^{\phi}_X)$ has a width-bounded decomposition relative to X and ϕ provided in the input, or (b) every $G_V(\widehat{M}^{\phi}_{X'})$ with $X' \subseteq X$ has a width-bounded decomposition relative to X' and ϕ provided in the input. This result essentially follows from Theorems 7.3 and 7.4. Note that in Theorems 7.5 to 7.9, each of (a) and (b) implies that $|X|$ is bounded by a constant, and thus also the number of all subsets $X' \subseteq X$ is bounded by a constant. Theorems 7.5 to 7.9 also hold, when the decompositions are relative to $X \cap R^{\phi}_X(M)$ and $X' \cap \widehat{R}^{\phi}_{X'}(M)$ rather than X and X', respectively.

Theorem 7.5. *Given a domain-bounded causal model $M = (U, V, F)$, $X \subseteq V$, $u \in D(U)$, an event ϕ, and either* (a) *a width-bounded decomposition $\mathcal{D}$ of the graph $G_V(M^{\phi}_X)$ relative to X and ϕ, or* (b) *for every $X' \subseteq X$, a width-bounded decomposition $\mathcal{D}_{X'}$ of $G_V(\widehat{M}^{\phi}_{X'})$ relative to X' and ϕ, computing the set of all $X' = x'$, where $X' \subseteq X$ and $x' \in D(X')$, such that $X' = x'$ is a weak (resp., an actual) cause of ϕ under u in M is possible in polynomial time.*

7.3. Deciding and computing explanations and partial explanations

We now turn to deciding and computing explanations and partial explanations. The following theorem shows that deciding whether $X = x$ is an explanation of ϕ relative to $\mathcal{C} \subseteq D(U)$ in M can be done in polynomial time, if we assume the same restrictions as in Theorem 7.5. This result follows from Theorems 7.3 and 7.4.

Theorem 7.6. *Given a domain-bounded causal model $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $\mathcal{C} \subseteq D(U)$, an event ϕ, and either* (a) *a width-bounded decomposition $\mathcal{D}$ of $G_V(M^{\phi}_X)$ relative to X and ϕ, or* (b) *for each $X' \subseteq X$, a width-bounded decomposition $\mathcal{D}_{X'}$ of $G_V(\widehat{M}^{\phi}_{X'})$ relative to X' and ϕ, deciding whether $X = x$ is an explanation of ϕ relative to $\mathcal{C}$ in M can be done in polynomial time.*

A similar tractability result holds for deciding whether $X = x$ is a partial or an α-partial explanation of ϕ relative to some $(\mathcal{C}, P)$ in M, and for computing the explanatory power of a partial explanation.

Theorem 7.7. *Given a domain-bounded causal model $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $\mathcal{C} \subseteq D(U)$, an event ϕ, where $\phi(u)$ for all $u \in \mathcal{C}$, a probability function P on $\mathcal{C}$, and either* (a) *a width-bounded decomposition $\mathcal{D}$ of $G_V(M_X^\phi)$ relative to X and ϕ, or* (b) *for every $X' \subseteq X$, a width-bounded decomposition $\mathcal{D}_{X'}$ of $G_V(\widehat{M}_{X'}^\phi)$ relative to X' and ϕ,*

(1) *deciding whether $X = x$ is a partial explanation of ϕ relative to $(\mathcal{C}, P)$ in M can be done in polynomial time*;
(2) *deciding whether $X = x$ is an α-partial explanation of ϕ relative to $(\mathcal{C}, P)$ in M, for some given $\alpha \geqslant 0$, can be done in polynomial time*;
(3) *given that $X = x$ is a partial explanation of ϕ relative to $(\mathcal{C}, P)$ in M, computing the explanatory power of $X = x$ can be done in polynomial time.*

Such tractability results also hold for computing explanations and partial explanations. In particular, the next theorem shows that computing all explanations involving variables from a given set of endogenous variables is tractable under the same assumptions as in Theorem 7.5.

Theorem 7.8. *Given a domain-bounded causal model $M = (U, V, F)$, $X \subseteq V$, $\mathcal{C} \subseteq D(U)$, an event ϕ, and either* (a) *a width-bounded decomposition $\mathcal{D}$ of $G_V(M_X^\phi)$ relative to X and ϕ, or* (b) *for every $X' \subseteq X$, a width-bounded decomposition $\mathcal{D}_{X'}$ of $G_V(\widehat{M}_{X'}^\phi)$ relative to X' and ϕ, computing the set of all $X' = x'$, where $X' \subseteq X$ and $x' \in D(X')$, such that $X' = x'$ is an explanation of ϕ relative to $\mathcal{C}$ in M can be done in polynomial time.*

Similarly, also computing all partial and α-partial explanations involving variables from a given set of endogenous variables is tractable under the same restrictions as in Theorem 7.5.

Theorem 7.9. *Given a domain-bounded causal model $M = (U, V, F)$, $X \subseteq V$, $\mathcal{C} \subseteq D(U)$, an event ϕ, where $\phi(u)$ for all $u \in \mathcal{C}$, a probability function P on $\mathcal{C}$, and either* (a) *a width-bounded decomposition $\mathcal{D}$ of $G_V(M_X^\phi)$ relative to X and ϕ, or* (b) *for every $X' \subseteq X$, a width-bounded decomposition $\mathcal{D}_{X'}$ of $G_V(\widehat{M}_{X'}^\phi)$ relative to X' and ϕ,*

(1) *computing the set of all $X' = x'$, where $X' \subseteq X$ and $x' \in D(X')$, such that $X' = x'$ is a partial explanation of ϕ relative to $(\mathcal{C}, P)$ in M can be done in polynomial time*;
(2) *computing the set of all $X' = x'$ where $X' \subseteq X$ and $x' \in D(X')$, such that $X' = x'$ is an α-partial explanation of ϕ relative to $(\mathcal{C}, P)$ in M, for some given $\alpha \geqslant 0$, can be done in polynomial time.*

7.4. Computing degrees of responsibility and blame

We now show that the tractability results for deciding and computing causes and explanations of Sections 7.2 and 7.3 can also be extended to computing degrees of responsibility and blame. To this end, we slightly generalize the relations R^i, $i \in \{0, \ldots, k\}$, of Section 7.1.

We use the following notation. For sets of variables X and values $x, x' \in D(X)$, the *difference between x and x'*, denoted $\mathrm{diff}(x, x')$, is the number of all variables $A \in X$ such that $x(A) \neq x'(A)$.

We define R^0 as follows:

$$\begin{aligned} R^0 = \big\{ (\boldsymbol{p}, \boldsymbol{q}, F, l) \mid\ & F \subseteq S^0, \boldsymbol{p}, \boldsymbol{q} \subseteq D(F), l \in \{0, \ldots, |T^0|\}, \\ & \exists W \subseteq T^0, F = S^0 \setminus W, \\ & \exists w \in D(W)\ \forall p, q \in D(F): \\ & l = \mathrm{diff}(w, W(u)), \quad p \in \boldsymbol{p} \text{ iff } \neg\phi_{pw}(u), \\ & q \in \boldsymbol{q} \text{ iff } \phi_{[q\langle \hat{Z}(u)]w'}(u) \text{ for all } W' \subseteq W, w' = w|W', \text{ and } \hat{Z} \subseteq T^0 \setminus (S^k \cup W) \big\}. \end{aligned}$$

For every $i \in \{1, \ldots, k\}$, we then define R^i as follows:

$$\begin{aligned} R^i = \{ (\boldsymbol{p}, \boldsymbol{q}, F, l) \mid\ & F \subseteq S^i, \boldsymbol{p}, \boldsymbol{q} \subseteq D(F), l \in \{0, \ldots, \textstyle\sum_{j=0}^{i} |T^j|\}, \\ & \exists W \subseteq T^i, F = S^i \setminus W, \end{aligned}$$

$$\exists w \in D(W)\ \exists(\boldsymbol{p}', \boldsymbol{q}', F', l') \in R^{i-1}\ \forall p, q \in D(F):$$
$$l = \mathrm{diff}(w, W(u)) + l', \quad p \in \boldsymbol{p} \text{ iff } {F'}_{pw}(u) \in \boldsymbol{p}',$$
$$q \in \boldsymbol{q} \text{ iff } {F'}_{[q(\hat{Z}(u)]w'}(u) \in \boldsymbol{q}' \text{ for all } W' \subseteq W, w' = w|W', \text{ and } \hat{Z} \subseteq T^i \setminus (S^k \cup W)\}.$$

Intuitively, rather than triples $(\boldsymbol{p}, \boldsymbol{q}, F)$, the new relations R^i contain tuples $(\boldsymbol{p}, \boldsymbol{q}, F, l)$, where $\boldsymbol{p}$ (resp., $\boldsymbol{q}$) is a set of possible values of $F \subseteq S^i$ in **AC2**(a) (resp., (b)) as in Section 7.1, and l is the sum of all differences between $w \in D(W)$ and $W(u)$ in T^j for all $j \in \{0, \ldots, i\}$. Thus, **AC2** holds with some $W \subseteq V \setminus X$ and $w \in D(W)$ such that $\mathrm{diff}(w, W(u)) = l$ iff some $(\boldsymbol{p}, \boldsymbol{q}, X, l) \in R^k$ exists such that $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$. This result is expressed by the following theorem.

Theorem 7.10. *Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$, let $x \in D(X)$ and $u \in D(U)$, and let ϕ be an event. Let $((T^0, S^0), \ldots, (T^k, S^k))$ be a decomposition of $G_V(M)$ relative to X and ϕ, and let R^k be defined as above. Then,* **AC2** *holds with some $W \subseteq V \setminus X$ and $w \in D(W)$ such that $\mathrm{diff}(w, W(u)) = l$ iff some $(\boldsymbol{p}, \boldsymbol{q}, X, l) \in R^k$ exists such that $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$.*

The next theorem shows that the degree of responsibility of $X = x$ for ϕ in a situation (M, u) with domain-bounded M can be computed in polynomial time given that $G_V(M)$ has a width-bounded decomposition provided in the input. It follows from Theorem 7.10 and the fact that recursively computing the R^i's and deciding whether there exists some $(\boldsymbol{p}, \boldsymbol{q}, X, l) \in R^k$ with $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$ can be done in polynomial time under the above boundedness assumptions.

Theorem 7.11. *Given a domain-bounded causal model $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $u \in D(U)$, an event ϕ, and a width-bounded decomposition $\mathcal{D}$ of $G_V(M)$ relative to X and ϕ, computing the degree of responsibility of $X = x$ for ϕ in (M, u) is possible in polynomial time.*

Similarly, computing the degree of blame relative to an epistemic state $(\mathcal{K}, P)$ is tractable, when every causal model in $\mathcal{K}$ satisfies the same boundedness assumptions as in Theorem 7.11. This is expressed by the following theorem.

Theorem 7.12. *Given an epistemic state $(\mathcal{K}, P)$, where for every $(M, u) \in \mathcal{K}$, M is domain-bounded, a set of endogenous variables X, a value $x \in D(X)$, an event ϕ, and for every $(M, u) = ((U, V, F), u) \in \mathcal{K}$ a width-bounded decomposition of $G_V(M)$ relative to X and ϕ, computing the degree of blame of setting X to x for ϕ relative to $(\mathcal{K}, P)$ is possible in polynomial time.*

7.5. Computing decompositions

The tractability results of Sections 7.2 to 7.4 are all based on the assumption that some decomposition of $G_V(M)$ is provided in the input. It thus remains to decide whether such decompositions exist at all, and if so, then to compute one, especially one of minimal width. The problem of deciding whether there exists some decomposition of width below a given integer $l > 0$ is formally expressed as follows.

LAYERWIDTH WITH CONSTRAINTS: Given $G_V(M)$ for $M = (U, V, F)$, $X \subseteq V$, an event ϕ, and an integer $l > 0$, decide whether there exists a decomposition $((T^0, S^0), \ldots, (T^k, S^k))$ of $G_V(M)$ relative to X and ϕ of width at most l.

As shown by Hopkins [22], LAYERWIDTH WITH CONSTRAINTS is NP-complete. Hopkins [22] also presents an algorithm for computing a layer decomposition of lowest width, where a *layer decomposition* satisfies every condition among **D1** to **D6** except for **D3**. It is an any-time depth-first branch-and-bound algorithm, which searches through a binary search tree that represents the set of all possible layer decompositions. This algorithm can also be used to compute the set of all decompositions of $G_V(M)$ relative to X and ϕ of lowest width.

The intractability of computing a decomposition of lowest width, which is a consequence of the NP-completeness of LAYERWIDTH WITH CONSTRAINTS, is not such a negative result as it might appear at first glance. It means that

decompositions are an expressive concept, for which sophisticated algorithms like Hopkin's are needed to obtain good performance. However, the effort for decomposition pays off by subsequent polynomial-time solvability of a number of reasoning tasks given that the ramifying conditions are met, such that overall, the effort is polynomial time modulo calls to an NP-oracle. This complexity is lower than the complexity of weak and actual causes, as well as the complexity of explanations in the general case, which are located at the second and the third level of the Polynomial Hierarchy, respectively [6,7] (see also Section 4.4). On the other hand, the lower complexity means that suitable decompositions will not always exist. However, the worst-case complexity results in [6,7] use quite artificial constructions, and the causal models involved will hardly occur in practice. In fact, many of the examples in the literature seem to have decomposable causal graphs; it remains to be seen whether this holds for a growing stock of applications.

8. Layered causal graphs

In general, as described in Section 7.5, causal graphs $G_V(M)$ with width-bounded decompositions cannot be efficiently recognized, and such decompositions also cannot be efficiently computed. But, from Section 6, we already know width-bounded causal trees as a large class of causal graphs, which have width-bounded decompositions that can be computed in linear time. In this section, we discuss an even larger class of causal graphs, called *layered causal graphs*, which also have natural nontrivial decompositions that can be computed in linear time. Intuitively, in layered causal graphs $G_V(M)$, the set of endogenous variables V can be partitioned into *layers* $S^0, \ldots, S^k$ such that every arrow in $G_V(M)$ goes from a variable in some layer S^i to a variable in S^{i-1} (see Fig. 11).

In particular, when dealing with structure-based causes and explanations in first-order reasoning about actions as described in [9], binary causal models with a large number of variables are generated in a grounding step from first-order theories in the ICL (such as the one given in Example 3.1), which have a natural layering through the time line, and thus are a special kind of layered causal graphs.

We now first define layered causal graphs. We then prove that they are a special case of decomposable causal graphs, and that recognizing them and computing their layers can be done in linear time. In the sequel, let $M = (U, V, F)$ be a causal model, let $X \subseteq V$, let $x \in D(X)$ and $u \in D(U)$, and let ϕ be an event.

More formally, a *layering* of $G_V(M)$ relative to X and ϕ is an ordered partition $(S^0, \ldots, S^k)$ of V that satisfies the following conditions **L1** and **L2**:

L1. For every arrow $A \rightarrow B$ in $G_V(M)$, there exists some $i \in \{1, \ldots, k\}$ such that $A \in S^i$ and $B \in S^{i-1}$.
L2. Every $A \in V$ occurring in ϕ belongs to S^0, and $S^k \supseteq X$.

We say that $G_V(M)$ is *layered* relative to X and ϕ iff it has a layering $(S^0, \ldots, S^k)$ relative to X and ϕ. The *width* of such a layering $(S^0, \ldots, S^k)$ is the maximum of all $|S^i|$ such that $i \in \{0, \ldots, k\}$. A layered causal graph $G_V(M)$ relative to X and ϕ is *width-bounded* for an integer $l \geqslant 0$ iff there exists a layering $(S^0, \ldots, S^k)$ of $G_V(M)$ relative to X and ϕ of a width of at most l.

Example 8.1. Fig. 12 provides a layering $\mathcal{L} = (S^0, S^1, S^2)$ of the causal graph $G_V(\widehat{M}_X^{\phi})$ in Fig. 6 relative to $X' = X \cap \widehat{R}_X^{\phi}(M)$ and ϕ, where $M = (U, V, F)$ is a causal model and ϕ is an event such that the causal graph $G_V(M)$ and the sets X and $V(\phi)$ are as in Fig. 2. The width of this layering $\mathcal{L}$ is given by 3.

The following result shows that layered causal graphs $G_V(M)$ relative to X and ϕ have a natural nontrivial decomposition relative to X and ϕ.

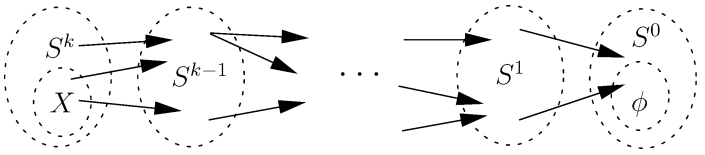

Fig. 11. Path from X to ϕ in a layered causal graph.

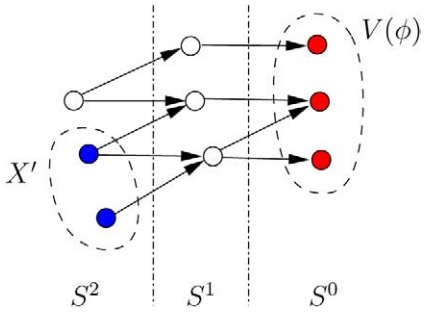

Fig. 12. Layering (S^0, S^1, S^2) of $G_V(\widehat{M}_X^{\phi})$ relative to $X' = X \cap \widehat{R}_X^{\phi}(M)$ and ϕ.

Proposition 8.1. *Let $M = (U, V, F)$ be a causal model, let $X \subseteq V$, and let ϕ be an event. Let $(S^0, \ldots, S^k)$ be a layering of $G_V(M)$ relative to X and ϕ. Then, $((S^0, S^0), \ldots, (S^k, S^k))$ is a decomposition of $G_V(M)$ relative to X and ϕ.*

We next give a condition under which a layered causal graph $G_V(M)$ has a unique layering. Two variables $A, B \in V$ are *connected* in $G_V(M)$ iff they are connected via a path in the undirected graph $(V, \{\{A, B\} \mid A \rightarrow B \text{ in } G_V(M)\})$. A causal graph $G_V(M)$ is *connected* relative to X and ϕ iff (i) $X \neq \emptyset$, (ii) there exists a variable in X that is connected to a variable in ϕ, and (iii) every variable in $V \setminus (X \cup V(\phi))$ is connected to a variable in $X \cup V(\phi)$. Notice that if $X = x$ is a weak cause of ϕ under $u \in D(U)$, then (i) and (ii) hold. Furthermore, in (iii), each variable $A \in V \setminus (X \cup V(\phi))$ which is not connected to a variable in $X \cup V(\phi)$ is irrelevant to "$X = x$ is a weak cause of ϕ under u".

The next result shows that when layered causal graphs $G_V(M)$ relative to X and ϕ are connected relative to X and ϕ, then the layering is unique. For this result, observe that every event ϕ contains some variables $A \in V$, which are all placed in the layer S_0. By conditions (i) and (ii), also X contains some variables, which are all placed in some layer S^k. By condition (iii), any other variable belongs to at most one layer S^i, and thus to exactly one layer S^i, since $G_V(M)$ is layered.

Proposition 8.2. *Let $M = (U, V, F)$ be a causal model, let $X \subseteq V$, and let ϕ be an event. If $G_V(M)$ is layered and connected relative to X and ϕ, then $G_V(M)$ has a unique layering relative to X and ϕ.*

We now provide an algorithm for deciding if a connected causal graph $G_V(M)$ relative to X and ϕ is layered and, if so, for computing its unique layering: Algorithm LAYERING (see Fig. 13) computes the unique layering $\mathcal{L} = (S^0, \ldots, S^k)$ of a connected causal graph $G_V(M)$ relative to X and ϕ, if it exists. The layering $\mathcal{L}$ is represented by the mapping $\lambda : V \rightarrow \{0, \ldots, k\}$, defined by $\lambda(A) = j$ for all $A \in S^j$ and all $j \in \{0, \ldots, k\}$. The following proposition states the correctness of LAYERING.

Proposition 8.3. *Let $M = (U, V, F)$ be a causal model, let $X \subseteq V$, and let ϕ be an event, where $G_V(M)$ is connected relative to X and ϕ. Then,* LAYERING *returns the unique layering of $G_V(M)$ relative to X and ϕ, if it exists, and Nil, otherwise.*

The next result shows that recognizing layered and width-bounded causal graphs $G_V(M)$ and computing their unique layerings can be done in linear time. Note that deciding whether $G_V(M)$ is connected w.r.t. X and ϕ is also possible in linear time.

Proposition 8.4. *Given a causal model $M = (U, V, F)$, $X \subseteq V$, and an event ϕ, where $G_V(M)$ is connected w.r.t. X and ϕ, deciding whether $G_V(M)$ is layered w.r.t. X and ϕ as well as width-bounded for some integer $l \geqslant 0$, and computing the unique layering of $G_V(M)$ w.r.t. X and ϕ can be done in $O(\|G_V(M)\| + \|\phi\|)$ time.*

By Proposition 8.1, all results of Sections 7.1–7.4 on causes, explanations, responsibility, and blame in decomposable causal graphs also apply to layered causal graphs as a special case. In particular, the relations R^i of Section 7.1 can be simplified to the following ones for layered causal graphs. The relation R^0 is given by:

Algorithm LAYERING

Input: causal model $M = (U, V, F)$, $X \subseteq V$, and an event ϕ,
where $G_V(M) = (V, E)$ is connected relative to X and ϕ.
Output: unique layering $\mathcal{L} = (S^0, \ldots, S^k)$ of $G_V(M)$ relative to X and ϕ,
if it exists; *Nil*, otherwise.
Notation: $\mathcal{L}$ is represented by the mapping $\lambda : V \to \{0, \ldots, k\}$, which is
defined by $\lambda(A) = j$ for all $A \in S^j$ and all $j \in \{0, \ldots, k\}$.

1. **for each** $A \in V \setminus V(\phi)$ **do** $\lambda(A) := \bot$ (i.e., *undefined*);
2. **for each** $A \in V \cap V(\phi)$ **do** $\lambda(A) := 0$;
3. **if** $X \cap V(\phi) \neq \emptyset$ **then for each** $A \in X$ **do** $\lambda(A) := 0$;
4. **while** $E \neq \emptyset$ **do begin**
5. select some $A \to B$ in E such that $\lambda(A) \neq \bot$ or $\lambda(B) \neq \bot$;
6. **if** $B \in X \vee \lambda(A) = 0$ **then return** *Nil*;
7. **if** $\lambda(A) \neq \bot \wedge \lambda(B) = \bot$ **then** $\lambda(B) := \lambda(A) - 1$
8. **else if** $\lambda(A) = \bot \wedge \lambda(B) \neq \bot$ **then begin** $\lambda(A) := \lambda(B) + 1$;
9. **if** $A \in X$ **then for each** $A' \in X \setminus \{A\}$ **do** $\lambda(A') := \lambda(A)$
10. **end**
11. **else** /* $\lambda(A), \lambda(b) \neq \bot$ */ **if** $\lambda(A) \neq \lambda(B) + 1$ **then return** *Nil*;
12. $E := E \setminus \{A \to B\}$
13. **end**;
14. **if** $X \subseteq \{A \in V \mid \lambda(A) = k\}$, where $k = \max\{\lambda(A) \mid A \in V\}$ **then return** λ
15. **else return** *Nil*.

Fig. 13. Algorithm LAYERING.

$$
\begin{aligned}
R^0 = \big\{ (\boldsymbol{p}, \boldsymbol{q}, F) \mid\ & F \subseteq S^0,\ \boldsymbol{p}, \boldsymbol{q} \subseteq D(F), \\
& \exists w \in D(S^0 \setminus F)\ \forall p, q \in D(F): \\
& p \in \boldsymbol{p} \text{ iff } \neg\phi_{pw}(u), \\
& q \in \boldsymbol{q} \text{ iff } \phi_{[q\langle \hat{Z}(u)]w'}(u) \text{ for all } W' \subseteq S^0 \setminus F,\ w' = w|W',\ \text{and } \hat{Z} \subseteq F \setminus S^k \big\}.
\end{aligned}
$$

For each $i \in \{1, \ldots, k\}$, the relation R^i is given by:

$$
\begin{aligned}
R^i = \big\{ (\boldsymbol{p}, \boldsymbol{q}, F) \mid\ & F \subseteq S^i,\ \boldsymbol{p}, \boldsymbol{q} \subseteq D(F), \\
& \exists w \in D(S^i \setminus F)\ \exists (\boldsymbol{p'}, \boldsymbol{q'}, F') \in R^{i-1}\ \forall p, q \in D(F): \\
& p \in \boldsymbol{p} \text{ iff } {F'}_{pw}(u) \in \boldsymbol{p'}, \\
& q \in \boldsymbol{q} \text{ iff } {F'}_{[q\langle \hat{Z}(u)]w'}(u) \in \boldsymbol{q'} \text{ for all } W' \subseteq S^i \setminus F,\ w' = w|W',\ \text{and } \hat{Z} \subseteq F \setminus S^k \big\}.
\end{aligned}
$$

The following theorem is immediate by Theorem 7.1 and Proposition 8.1.

Theorem 8.5. *Let $M = (U, V, F)$ be a causal model. Let $X \subseteq V$, let $x \in D(X)$ and $u \in D(U)$, and let ϕ be an event. Let $(S^0, \ldots, S^k)$ be a layering of $G_V(M)$ relative to X and ϕ, and let R^k be defined as above. Then, $X = x$ is a weak cause of ϕ under u in M iff (α) $X(u) = x$ and $\phi(u)$ in M, and (β) some $(\boldsymbol{p}, \boldsymbol{q}, X) \in R^k$ exists such that $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$.*

The next theorem shows that deciding whether $X = x$ is a weak respectively actual cause of ϕ under u in domain-bounded M is tractable, when $G_V(M)$ is layered and width-bounded. This is immediate by Theorem 7.2 and Proposition 8.1.

Theorem 8.6. *Given a domain-bounded causal model $M = (U, V, F)$, $X \subseteq V$, $x \in D(X)$, $u \in D(U)$, and an event ϕ, where $G_V(M)$ is layered (relative to X and ϕ) and width-bounded for a constant $l \geqslant 0$, deciding whether $X = x$ is a weak (resp., an actual) cause of ϕ under u in M is possible in polynomial time.*

Similarly, by Proposition 8.1, all the tractability results of Theorems 7.3–7.9 and Theorems 7.11 and 7.12 also hold for width-bounded layered causal graphs.

9. Extended causal models

In this section, we show that with some slight technical adaptations, all the above techniques and results carry over to a recent generalization of causal models to extended causal models due to Halpern and Pearl [17]. This shows that the results are robust at their core.

An *extended causal model* $M = (U, V, F, E)$ consists of a standard causal model (U, V, F) as in Section 2.1 and a set $E \subseteq D(V)$ of *allowable settings* for V. For any $Y \subseteq V$, a value $y \in D(Y)$ is an *allowable setting* for Y iff $y = v|Y$ for some $v \in E$. Informally, y can be extended to an allowable setting for V. In the sequel, we assume E is represented in such a way that deciding whether a given $y \in D(Y)$, $Y \subseteq V$, is an allowable setting for Y is possible in polynomial time.

The notion of *weak cause* in extended causal models $M = (U, V, F, E)$ is then defined by slightly modifying the conditions **AC2**(a) and **AC2**(b) in the definition of weak causality to restrict to allowable settings.

To extend the results in Section 5.1 to the notion of weak cause in extended causal models, we introduce a natural closure property as follows. We say $M = (U, V, F, E)$ is *closed* (resp., *closed relative to* $X \subseteq V$) iff $y \cup (V \setminus Y)_y(u)$ is an allowable setting for V for all allowable settings y for Y, all $Y \subseteq V$ (resp., all $Y \subseteq V$ with $X \subseteq Y$), and all $u \in D(U)$. Informally, if y is an allowable setting for Y, then extending y by the values of all other endogenous variables in M_y under any $u \in D(U)$ is an allowable setting for V. Notice that M is closed relative to all $X \subseteq V$, if M is closed. The following result says that Theorems 5.1 and 5.2 carry over to the notion of weak cause in closed extended causal models.

Theorem 9.1. *Theorems* 5.1 *and* 5.2 *hold also for the notion of weak cause in extended causal models* $M = (U, V, F, E)$ *that are closed relative to* X'.

For the results of Sections 5.2 and 5.3, we generalize the notions of a reduction and a strong reduction to extended causal models as follows. The *reduced* (resp., *strongly reduced*) *extended causal model* of $M = (U, V, F, E)$ w.r.t. $X = x$ and ϕ, denoted $M^{\phi}_{X=x}$ (resp., $\widehat{M}^{\phi}_{X=x}$), is defined as the extended causal model $M' = (U, V', F', E')$, where (U, V', F') is the reduced (resp., strongly reduced) causal model of (U, V, F) w.r.t. $X = x$ and ϕ, and $E' = \{v|V' \mid v \in E\}$. Notice here that, since E' is the restriction of E to V', any procedure for deciding allowability relative to E is immediately a procedure for deciding allowability relative to E'. The following result says that reductions and strong reductions keep the closure property.

Theorem 9.2. *Let* $M = (U, V, F, E)$ *be an extended causal model. Let* $X \subseteq V$ *and* $x \in D(X)$, *let* $X' = X \cap \widehat{R}^{\phi}_X(M)$, *and let* ϕ *be an event. Then*: (a) *if* M *is closed, then also* M^{ϕ}_X *is closed*; (b) *if* M *is closed relative to* X', *then also* $\widehat{M}^{\phi}_X$ *is closed relative to* X'.

Using these notations, all the results of Sections 5.2 and 5.3 hold also for the notion of weak cause in closed extended causal models. In particular, the following theorem generalizes Theorems 5.7 and 5.10.

Theorem 9.3. *Let* $M = (U, V, F, E)$ *be an extended causal model. Let* $X' \subseteq X \subseteq V$ (*resp.*, $X' = X \subseteq V$), *let* $x' \in D(X')$, $x \in D(X)$, *and* $u \in D(U)$, *and let* ϕ *be an event. Let* $X'' = X' \cap R^{\phi}_X(M)$ (*resp.*, $X'' = X' \cap \widehat{R}^{\phi}_X(M)$) *and* $x'' = x'|X''$. *Suppose that* M *is closed relative to* X''. *Then,* $X' = x'$ *is a weak cause of* ϕ *under* u *in* M *iff* (i) $(X' \setminus X'')(u) = x'|(X' \setminus X'')$ *in* M, *and* (ii) $X'' = x''$ *is a weak cause of* ϕ *under* u *in* M^{ϕ}_X (*resp.*, $\widehat{M}^{\phi}_X$).

For the results of Sections 7.1 to 7.3, we generalize the notion of a decomposition of $G_V(M)$ and the relations R^i, $i \in \{0, \ldots, k\}$, in Section 7.1 to extended causal models as follows. A *decomposition* of $G_V(M)$ relative to $X = x$ (or simply X) and ϕ is a tuple $((T^0, S^0), \ldots, (T^k, S^k))$, $k \geqslant 0$, of pairs (T^i, S^i) such that **D1**–**D6** in Section 7.1 and the following condition **D7** hold:

D7. Every y^i with $i \in \{0, \ldots, k\}$ is an allowable setting of $Y^i \subseteq T^i$ iff $y^0 \cup \cdots \cup y^k$ is an allowable setting of $Y^0 \cup \cdots \cup Y^k \subseteq V$.

We then finally adapt the relations R^i, $i \in \{0, \ldots, k\}$, in Section 7.1 by replacing "$\neg\phi_{pw}(u)$" and "$F'_{pw}(u) \in \boldsymbol{p'}$" with "$\neg\phi_{pw}(u)$ and $pw|(X \cup W)$ is allowable" and "$F'_{pw}(u) \in \boldsymbol{p'}$ and $pw|(X \cup W)$ is allowable", respectively.

Using the above notations, all the results of Sections 7.1 to 7.3 (and thus all the results of Sections 6 and 8) hold also for the notion of weak cause in closed extended causal models. In particular, the following theorem is a generalization of Theorem 7.1 to the notion of weak cause in closed extended causal models. Note that the results of Section 7.4 can be similarly generalized.

Theorem 9.4. *Let $M = (U, V, F, E)$ be an extended causal model. Let $X \subseteq V$, let $x \in D(X)$ and $u \in D(U)$, and let ϕ be an event. Let $((T^0, S^0), \ldots, (T^k, S^k))$ be a decomposition of $G_V(M)$ relative to X and ϕ, and let R^k be defined as above. Suppose that M is closed relative to X. Then, $X = x$ is a weak cause of ϕ under u in M iff (α) $X(u) = x$ and $\phi(u)$ in M, and (β) some $(\boldsymbol{p}, \boldsymbol{q}, X) \in R^k$ exists such that $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$.*

10. Conclusion

Defining causality between events is an issue which beyond the philosophical literature has also been considered in AI for a long time. Because of its key role for hypothesis and explanation forming, it is an important problem for which a number of different approaches have been proposed. In the approach by Halpern and Pearl [15–17], causality is modeled using structural equations, distinguishing between weak and actual causes of events which are modeled by Boolean combinations of atomic value statements. Based on weak causes, a notion of causal explanation and probabilistic variants thereof have been defined in [18], while a refinement of actual causality in terms of responsibility and blame has been recently given in [3]. As has been argued and demonstrated, the structural-model approach by Halpern and Pearl deals well with difficulties of other approaches, including recent ones in the literature (see [16–18]).

In order to bring the theoretical approach by Halpern and Pearl to practice, an understanding of the computational properties and (efficient) algorithms are required. In this direction, the computational complexity of major decision and computation problems for the approach has been studied in [3,6,7], and algorithms for computing causes proposed in [21]. Since arbitrary Boolean functions are used to model structural equations, determining causes and explanations is unsurprisingly intractable in general. Hence, the important issue of tractable cases arose, as well as how unnecessary complexity in computations can be avoided.

Investigating these issues, we have first explored, extending work by Hopkins [21], how to focus the computation of (potential) weak causes and causal models, by efficient removal of irrelevant variables. We have then presented a new characterization of weak cause for a certain class of causal models in which the causal graph over the endogenous variables is benignly decomposable. Two natural and important subclasses of it are causal trees and layered causal graphs, which can be efficiently recognized, namely in linear time. By combining the removal of irrelevant variables with this new characterization of weak cause, we have then obtained techniques for deciding and computing causes and explanations in the structural-model approach, which show that these problems are tractable under suitable conditions. To our knowledge, these are the first explicit tractability results for causes and explanations in the structural-model approach. Moreover, by slightly extending the new characterization of weak cause, we have obtained similar techniques for computing the degrees of responsibility and blame, and thus also new tractability results for structure-based responsibility and blame. Finally, we have shown that all the above techniques and results carry over to recent refinements of the notion of weak cause and causal models due to Halpern and Pearl [17], and we have also described an application of our results and techniques for dealing with structure-based causes and explanations in first-order reasoning about actions in Poole's ICL.

We have thus identified tractable special cases for decision and optimization problems of relatively high complexity, which is to some extend remarkable. These tractability results are a nice computational property of causes, explanations, responsibility, and blame in the structural-model approach.

An interesting topic of further studies is to explore whether there are other important classes of causal graphs different from causal trees and layered causal graphs in which the tractable cases can be recognized efficiently (that is, in which width-bounded decompositions can be recognized and computed efficiently). Another direction is analyzing how the techniques and results of this paper can be further developed for reasoning about ac-

tions [9] and commonsense causal reasoning [24]. Finally, implementation and further optimization remains for future work.

Acknowledgements

This work has been partially supported by the Austrian Science Fund under project Z29-N04, by a grant of the German Research Foundation, by a Marie Curie Individual Fellowship of the EU programme "Human Potential" under contract number HPMF-CT-2001-001286 (disclaimer: The authors are solely responsible for information communicated and the European Commission is not responsible for any views or results expressed), and by a Heisenberg Professorship of the German Research Foundation. We are thankful to the reviewers of this paper and its UAI-2002 abstract, whose constructive comments helped to improve our work.

Appendix A. Proofs for Section 5

Proof of Theorem 5.1 (resp., 5.2). Let $X_0 \in X$ be such that (α) there is no directed path in $G(M)$ from X_0 to a variable in ϕ (resp., (β) each directed path in $G(M)$ from X_0 to a variable in ϕ contains some $X_i \in X \setminus \{X_0\}$). Let $X'' = X \setminus \{X_0\}$ and $x'' = x|X''$. It is now sufficient to show that $X = x$ is a weak cause of ϕ under u iff (i) $X_0(u) = x(X_0)$ and (ii) $X'' = x''$ is a weak cause of ϕ under u.

($\Rightarrow$) Suppose that $X = x$ is a weak cause of ϕ under u. That is, **AC1** $X(u) = x$ and $\phi(u)$, and **AC2** some $W \subseteq V \setminus X$, $\bar{x} \in D(X)$, and $w \in D(W)$ exist such that (a) $\neg\phi_{\bar{x}w}(u)$ and (b) $\phi_{xw'\hat{z}}(u)$ for all $W' \subseteq W$, $\hat{Z} \subseteq V \setminus (X \cup W)$, $w' = w|W'$, and $\hat{z} = \hat{Z}(u)$. In particular, (i) $X_0(u) = x(X_0)$, and also **AC1** $X''(u) = x''$ and $\phi(u)$. By (α) (resp., (β)), it then follows that **AC2**(a) $\neg\phi_{\bar{x}''w}(u)$ and (b) $\phi_{x''w'\hat{z}}(u)$ hold for all $W' \subseteq W$, $\hat{Z} \subseteq V \setminus (X'' \cup W)$, $w' = w|W'$, and $\hat{z} = \hat{Z}(u)$, where $\bar{x}'' = \bar{x}|X''$. This shows that (ii) $X'' = x''$ is a weak cause of ϕ under u.

($\Leftarrow$) Suppose that (i) $X_0(u) = x(X_0)$ and (ii) $X'' = x''$ is a weak cause of ϕ under u. That is, **AC1** $X''(u) = x''$ and $\phi(u)$, and **AC2** some $W \subseteq V \setminus X''$, $\bar{x}'' \in D(X'')$, and $w \in D(W)$ exist such that (a) $\neg\phi_{\bar{x}''w}(u)$, and (b) $\phi_{x''w'\hat{z}}(u)$ for all $W' \subseteq W$, $\hat{Z} \subseteq V \setminus (X'' \cup W)$, $w' = w|W'$, and $\hat{z} = \hat{Z}(u)$. By (i), it thus holds **AC1** $X(u) = x$ and $\phi(u)$. By (α) (resp., (β)), it follows that **AC2**(a) $\neg\phi_{\bar{x}''\bar{x}_0w'}(u)$ and (b) $\phi_{x''x_0w''\hat{z}}(u)$ for all $W'' \subseteq W'$, $\hat{Z} \subseteq V \setminus (X \cup W')$, $w'' = w'|W''$, and $\hat{z} = \hat{Z}(u)$, where $W' = W \setminus \{X_0\}$, $w' = w|W'$, $\bar{x}_0 = (X_0)_{\bar{x}''w}(u)$, and $x_0 = x(X_0)$. This shows that $X = x$ is a weak cause of ϕ under u. □

Proof of Proposition 5.3. (a) We first compute the set A_ϕ of all variables in ϕ and their ancestors in $G_V(M)$, and then the set $X' = A_\phi \cap X$. Using standard methods and data structures, the first step can be done in time $O(\|\phi\| + |E|)$ where $G_V(M) = (V, E)$, and the second step in time $O(|V|)$. In summary, X' is computable in time $O(\|G_V(M)\| + \|\phi\|)$, and hence in time $O(\|M\| + \|\phi\|)$.

(b) We first compute the directed graph G' obtained from $G_V(M) = (V, E)$ by removing every arrow $X_k \to X_l$ with $X_l \in X$. We then compute the set ${A'}_\phi$ of all ancestors in G' of variables in ϕ. We finally compute $X' = {A'}_\phi \cap X$. Using standard methods and data structures, the first step can be done in time $O(|V| + |E|)$, the second step in time $O(|V| + |E| + \|\phi\|)$, and the third step in time $O(|V|)$. In summary, X' is computable in time $O(|V| + |E| + \|\phi\|)$, hence in time $O(\|M\| + \|\phi\|)$. □

Proof of Proposition 5.4. Recall that $R^\phi_X(M)$ is the set of all variables $A \in V$ from which there exists a directed path in $G(M)$ to a variable in ϕ. It thus follows immediately that $X \cap R^\phi_X(M)$ is the set of all variables $B \in X$ from which there exists a directed path in $G(M)$ to a variable in ϕ. □

Proof of Theorem 5.5. Let $M' = M^\phi_X = (U, V', F')$. Let $X' = X \cap V'$ and $x' = x|X'$. We have to show that $X = x$ is a weak cause of ϕ under u in M iff (i) $(X \setminus X')(u) = x|(X \setminus X')$ in M, and (ii) $X' = x'$ is a weak cause of ϕ under u in M^ϕ_X. The subsequent proof of this equivalence makes use of the following fact:

Fact A. *$V'_M(u) = V'_{M^\phi_X}(u)$ and $\phi_M(u) = \phi_{M^\phi_X}(u)$.*

($\Rightarrow$) Suppose that $X = x$ is a weak cause of ϕ under u in M. That is, **AC1** $X(u) = x$ and $\phi(u)$ in M, and **AC2** some $W \subseteq V \setminus X$, $\overline{x} \in D(X)$, $w \in D(W)$ exist such that (a) $\neg\phi_{\overline{x}w}(u)$ in M and (b) $\phi_{xw'\hat{z}}(u)$ in M for all $W' \subseteq W$, $\hat{Z} \subseteq V \setminus (X \cup W)$, $w' = w|W'$, and $\hat{z} = \hat{Z}(u)$ in M. This shows that (i) $(X \setminus X')(u) = x|(X \setminus X')$ in M. We next show that also (ii) holds. By Fact A, **AC1** $X'(u) = x'$ and $\phi(u)$ in M^{ϕ}_{X}. Notice then that (a) $\neg\phi_{\overline{x}'\overline{w}}(u)$ in M and (b) $\phi_{x'\overline{w}'\hat{z}'}(u)$ in M, where $\overline{x}' = \overline{x}|X'$, $\overline{W} = W \cap V'$, $\overline{w} = w|\overline{W}$, $\overline{W}' = W' \cap V'$, $\overline{w}' = w'|\overline{W}' = \overline{w}|\overline{W}'$, $\hat{Z}' = \hat{Z} \cap V'$, and $\hat{z}' = \hat{z}|\hat{Z}'$. Since each among $\neg\phi_{\overline{x}'\overline{w}}(u)$, $\phi_{x'\overline{w}'\hat{z}'}(u)$, and $\hat{Z}'(u)$ has the same values in M and M^{ϕ}_{X}, this shows that **AC2**(a) $\neg\phi_{\overline{x}'\overline{w}}(u)$ in M^{ϕ}_{X} and (b) $\phi_{x'\overline{w}'\hat{z}'}(u)$ in M^{ϕ}_{X} for all $\hat{Z}' \subseteq V' \setminus (X' \cup \overline{W})$, $\overline{W}' \subseteq \overline{W}$, $\overline{w}' = \overline{w}|\overline{W}'$, and $\hat{z}' = \hat{Z}'(u)$ in M^{ϕ}_{X}. In summary, (ii) $X' = x'$ is a weak cause of ϕ under u in M^{ϕ}_{X}.

($\Leftarrow$) Suppose that (i) $(X \setminus X')(u) = x|(X \setminus X')$ in M and (ii) $X' = x'$ is a weak cause of ϕ under u in M^{ϕ}_{X}. Thus, **AC1** $X'(u) = x'$ and $\phi(u)$ in M^{ϕ}_{X}, and **AC2** some $W \subseteq V' \setminus X'$, $\overline{x}' \in D(X')$, $w \in D(W)$ exist such that (a) $\neg\phi_{\overline{x}'w}(u)$ in M^{ϕ}_{X} and (b) $\phi_{x'w'\hat{z}}(u)$ in M^{ϕ}_{X} for all $W' \subseteq W$, $\hat{Z} \subseteq V' \setminus (X' \cup W)$, $w' = w|W'$, and $\hat{z} = \hat{Z}(u)$ in M^{ϕ}_{X}. By Fact A, **AC1** $X(u) = x$ and $\phi(u)$ in M. Since each among $\neg\phi_{\overline{x}'w}(u)$, $\phi_{x'w'\hat{z}}(u)$, and $\hat{Z}(u)$ has the same values in M and M^{ϕ}_{X}, this shows that (a) $\neg\phi_{\overline{x}'w}(u)$ in M and (b) $\phi_{x'w'\hat{z}}(u)$ in M for all $W' \subseteq W$, $\hat{Z} \subseteq V' \setminus (X' \cup W)$, $w' = w|W'$, and $\hat{z} = \hat{Z}(u)$ in M. It then follows that **AC2**(a) $\neg\phi_{\overline{x}w}(u)$ in M and (b) $\phi_{xw'\hat{z}}(u)$ in M for all $W' \subseteq W$, $\hat{Z} \subseteq V \setminus (X \cup W)$, $w' = w|W'$, and $\hat{z} = \hat{Z}(u)$ in M, where $\overline{x}|X' = \overline{x}'$ and $\overline{x}|(X \setminus X') = (X \setminus X')_{\overline{x}w}(u)$ in M. In summary, this shows that $X = x$ is a weak cause of ϕ under u in M. $\square$

Proof of Proposition 5.6. Let $F = \{F_A \mid A \in V\}$, $M^{\phi}_{X} = (U, V', F')$, and $M^{\phi}_{X'} = (U, V'', F'')$. We first show that $V' = V''$ and then that $F' = F''$, which proves that M^{ϕ}_{X} coincides with $M^{\phi}_{X'}$. The former follows from the fact that V' and V'' are both given by the set of all variables $A \in V$ from which there exists a directed path in $G(M)$ to a variable in ϕ, while the latter follows from the former and the fact that $F' = \{F_A \mid A \in V'\}$ and $F'' = \{F_A \mid A \in V''\}$. $\square$

Proof of Theorem 5.7. Let $X'' = X' \cap R^{\phi}_{X'}(M)$. By Theorem 5.5, $X' = x'$ is a weak cause of ϕ under u in M iff (i) $(X' \setminus X'')(u) = x'|(X' \setminus X'')$ in M, and (ii) $X'' = x''$ is a weak cause of ϕ under u in $M^{\phi}_{X'}$. By Proposition 5.6, it holds that $R^{\phi}_{X'}(M) = R^{\phi}_{X}(M)$ and $M^{\phi}_{X'} = M^{\phi}_{X}$, which proves the result. $\square$

Proof of Proposition 5.8. We first show that the directed graph $G_V(M^{\phi}_{X})$ is computable in linear time. Its set of nodes $V' = R^{\phi}_{X}(M)$ is the set of all variables $A \in V$ from which there exists a directed path in $G(M)$ to a variable in ϕ, which is computable in time $O(\|G(M)\| + \|\phi\|)$ and thus in time $O(\|M\| + \|\phi\|)$, using standard methods and data structures. This already shows that $G_V(M^{\phi}_{X})$ can be computed in time linear in the size of M and ϕ, since it is the restriction of $G(M)$ to V'.

We next show that also $M^{\phi}_{X} = (U, V', F')$ can be computed in linear time. Let $F = \{F_A \mid A \in V\}$. As argued above, V' can be computed in time $O(\|M\| + \|\phi\|)$, but since $F' = \{F_A \mid A \in V'\}$, also M^{ϕ}_{X} can be computed in time $O(\|M\| + \|\phi\|)$. $\square$

Proof of Proposition 5.9. Recall that $\widehat{R}^{\phi}_{X}(M)$ is the set of all $A \in V$ from which there exists a directed path in $G(M)$ to a variable in ϕ that contains no $X_j \in X \setminus \{A\}$. It thus follows directly that $X \cap \widehat{R}^{\phi}_{X}(M)$ is the set of all $X_i \in X$ from which there exists a directed path in $G(M)$ to a variable in ϕ that contains no $X_j \in X \setminus \{X_i\}$. $\square$

Proof of Theorem 5.10. The proof is similar to the one of Theorem 5.5, using the strongly reduced causal model $\widehat{M}^{\phi}_{X}$ instead of the reduced causal model M^{ϕ}_{X}. $\square$

Proof of Proposition 5.11. We first show that the directed graph $G_V(\widehat{M}^{\phi}_{X})$ is computable in linear time. Its set of nodes $V' = \widehat{R}^{\phi}_{X}(M)$ is the set of all variables in ϕ and of all ancestors in G' of variables in ϕ, where the directed graph G' is obtained from $G_V(M)$ by removing every arrow $X_k \rightarrow X_l$ with $X_l \in X$. Notice then that G' is computable in time $O(\|G_V(M)\|)$, and that V' is computable in time $O(\|G'\| + \|\phi\|)$, and thus in time $O(\|M\| + \|\phi\|)$, once G' is

given. In summary, V' is computable in time $O(\|M\| + \|\phi\|)$. This shows that $G_V(\widehat{M}_X^{\phi})$ is computable in time linear in the size of M and ϕ, since it is the restriction of $G(M)$ to V'.

We next show that $\widehat{M}_X^{\phi} = (U, V', F')$ is computable in polynomial time. Let $F = \{F_A \mid A \in V\}$. As argued above, V' can be computed in time $O(\|M\| + \|\phi\|)$. Recall then that $F' = \{F_A \mid A \in V' \setminus X\} \cup \{F_A^{\star} \mid A \in V' \cap X\}$. Since all set operations are feasible in linear time using standard methods and data structures, it is sufficient to show that a representation of every function $F_A^{\star}$, where $A \in V' \cap X$, is computable in time $O(\|M\|)$. Every $F_A^{\star}(U_A)$ is given as follows. The set of arguments U_A is the set of all ancestors $B \in U$ of A in $G(M)$. The function $F_A^{\star}$ itself can be represented by the restriction M_A of $M = (U, V, F)$ to V and all ancestors $B \in U$ of A in $G(M)$. Then, $F_A^{\star}(u_A)$ for $u_A \in D(U_A)$ is given by $A(u_A)$ in M_A. Observe that by Proposition 2.1, every $F_A^{\star}(u_A)$ is computable in polynomial time. Clearly, U_A and M_A can be computed in linear time. Thus, the set of all functions $F_A^{\star}$, where $A \in V' \cap X$, can be computed in time $O(|V|\|M\|)$. In summary, this shows that $\widehat{M}_X^{\phi} = (U, V', F')$ can be computed in time $O(|V|\|M\| + \|\phi\|)$. □

Appendix B. Proofs for Section 6

Proof of Proposition 6.1. Using standard methods and data structures, deciding whether there exists exactly one directed path in $G_V(M) = (V, E)$ from every variable $A \in V \setminus \{Y\}$ to Y can be done in $O(|V| + |E|)$ time. Moreover, deciding whether every $A \in V \setminus \{X\}$ has a bounded number of parents can also be done in $O(|V| + |E|)$ time. In summary, deciding whether M is a causal tree with respect to X and Y is feasible in $O(|V| + |E|) = O(\|M\|)$ time. By Proposition 5.8, the directed graph $G_V(\widehat{M}_X^{Y})$ is computable in $O(\|M\|)$ time from M and X, Y. Thus, deciding whether $\widehat{M}_X^{Y}$ is a (bounded) causal tree is also possible in time $O(\|M\|)$. □

Proof of Theorem 6.2. Clearly, (α) coincides with **AC1**. Assume that (α) holds. We now show that (β) is equivalent to **AC2**:

AC2. Some set of variables $W \subseteq V \setminus X$ and some values $\bar{x} \in D(X)$ and $w \in D(W)$ exist such that:
(a) $Y_{\bar{x}w}(u) \neq y$,
(b) $Y_{xw'\hat{Z}(u)}(u) = y$ for all $W' \subseteq W$, $w' = w|W'$, and $\hat{Z} \subseteq V \setminus (X \cup W)$.

Clearly, we can assume that $P^i \notin W$ for all $i \in \{0, \ldots, k-1\}$, since otherwise $Y_{\bar{x}w}(u) = Y_{xw}(u)$. This shows that $W \subseteq W^1 \cup \cdots \cup W^k$. Observe then that we can enlarge every $w \in D(W)$ to some $w^{\star} \in D(W^{\star})$, where $W^{\star} = W^1 \cup \cdots \cup W^k$, by defining $w^{\star}|W = w$ and $w^{\star}|(W^{\star} \setminus W) = (W^{\star} \setminus W)(u)$. Hence, we can assume that $\hat{Z} \subseteq \{P^0, \ldots, P^{k-1}\}$ and thus also, by the path structure of the causal tree, that $\hat{Z} = \{P^i\}$ with $i \in \{1, \ldots, k-1\}$. Hence, it is sufficient to prove that (β) is equivalent to the following condition **AC2***:

AC2*. Some values $\bar{x} \in D(X)$ and $\overline{w} \in D(\overline{W})$, where $\overline{W} = W^1 \cup \cdots \cup W^k$, exist such that:
(a) $Y_{\bar{x}\overline{w}}(u) \neq y$,
(b) $Y_{\hat{p}^j\overline{w}'}(u) = y$ for all $\overline{W}' \subseteq \overline{W}$, $\overline{w}' = \overline{w}|\overline{W}'$, and $j \in \{1, \ldots, k\}$.

We now show that $(\star)$ for every $i \in \{1, \ldots, k\}$, it holds that $(\boldsymbol{p}, \boldsymbol{q}) \in R^i$ iff there exists some $\overline{w} \in D(\overline{W})$, where $\overline{W} = W^1 \cup \cdots \cup W^i$, such that for all $p, q \in D(P^i)$:

(i) $p \in \boldsymbol{p}$ iff $Y_{p\overline{w}}(u) \neq y$,
(ii) $q \in \boldsymbol{q}$ iff $Y_{q\overline{w}'}(u), Y_{\hat{p}^j\overline{w}'}(u) = y$ for all $\overline{W}' \subseteq \overline{W}$, $\overline{w}' = \overline{w}|\overline{W}'$, and $j \in \{1, \ldots, \min(i, k-1)\}$.

This then shows that (β) is equivalent to **AC2***: $(\Rightarrow)$ Suppose that some $(\boldsymbol{p}, \boldsymbol{q}) \in R^k$ exists such that $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$. Then, some $\overline{w} \in D(\overline{W})$, where $\overline{W} = W^1 \cup \cdots \cup W^k$, and $p \in \boldsymbol{p}$ exist such that (a) $Y_{p\overline{w}}(u) \neq y$ and (b) $Y_{x\overline{w}'}(u), Y_{\hat{p}^j\overline{w}'}(u) = y$ for all $\overline{W}' \subseteq \overline{W}$, $\overline{w}' = \overline{w}|\overline{W}'$, and $j \in \{1, \ldots, k-1\}$. That is, **AC2*** holds. $(\Leftarrow)$ Suppose that **AC2***(a) and (b) hold for some $\bar{x} \in D(X)$ and $\overline{w} \in D(\overline{W})$, where $\overline{W} = W^1 \cup \cdots \cup W^k$. Let $\boldsymbol{p}$ (resp., $\boldsymbol{q}$) be the set of all $p \in D(P^k)$ (resp., $q \in D(P^k)$) such that $Y_{p\overline{w}}(u) \neq y$ (resp., $Y_{q\overline{w}'}(u), Y_{\hat{p}^j\overline{w}'}(u) = y$ for all $\overline{W}' \subseteq \overline{W}$, $\overline{w}' = \overline{w}|\overline{W}'$, and $j \in \{1, \ldots, k-1\}$). Then, $(\boldsymbol{p}, \boldsymbol{q}) \in R^k$, $\boldsymbol{p} \neq \emptyset$, and $x \in \boldsymbol{q}$. That is, (β) holds.

We prove ($\star$) by induction on $i \in \{1, \ldots, k\}$:

Basis: Since $R^0 = \{(D(Y) \setminus \{y\}, \{y\})\}$, it holds that $(\boldsymbol{p}, \boldsymbol{q}) \in R^1$ iff some $w \in D(W^1)$ exists such that for all $p, q \in D(P^1)$:

(i) $p \in \boldsymbol{p}$ iff $P^0_{pw}(u) \in D(Y) \setminus \{y\}$ iff $Y_{pw}(u) \neq y$,
(ii) $q \in \boldsymbol{q}$ iff $P^0_{qw'}(u), P^0_{\hat{p}^1 w'}(u) \in \{y\}$ iff $Y_{qw'}(u), Y_{\hat{p}^j w'}(u) = y$, for all $W' \subseteq W^1$, $w' = w|W'$, and $j \in \{1, \ldots, \min(1, k-1)\}$.

Induction: Recall that $(\boldsymbol{p}, \boldsymbol{q}) \in R^i$ iff some $w \in D(W^i)$ and $(\boldsymbol{p'}, \boldsymbol{q'}) \in R^{i-1}$ exist such that for all $p, q \in D(P^i)$:

(i′) $p \in \boldsymbol{p}$ iff $P^{i-1}_{pw}(u) \in \boldsymbol{p'}$,
(ii′) $q \in \boldsymbol{q}$ iff $P^{i-1}_{qw'}(u), P^{i-1}_{\hat{p}^j w'}(u) \in \boldsymbol{q'}$, for all $W' \subseteq W^i$, $w' = w|W'$, $j \in \{1, \ldots, \min(i, k-1)\}$.

By the induction hypothesis, $(\boldsymbol{p'}, \boldsymbol{q'}) \in R^{i-1}$ iff some $\overline{w}' \in D(\overline{W}')$, where $\overline{W}' = W^1 \cup \cdots \cup W^{i-1}$, exists such that for all $p', q' \in D(P^{i-1})$:

(i″) $p' \in \boldsymbol{p'}$ iff $Y_{p'\overline{w}'}(u) \neq y$,
(ii″) $q' \in \boldsymbol{q'}$ iff $Y_{q'\overline{w}''}(u), Y_{\hat{p}^j \overline{w}''}(u) = y$ for all $\overline{W}'' \subseteq \overline{W}'$, $\overline{w}'' = \overline{w}'|\overline{W}''$, and $j \in \{1, \ldots, \min(i-1, k-1)\}$.

Hence, $(\boldsymbol{p}, \boldsymbol{q}) \in R^i$ iff some $w \in D(W^i)$ and $\overline{w}' \in D(\overline{W}')$, where $\overline{W}' = W^1 \cup \cdots \cup W^{i-1}$, exist such that for all $p, q \in D(P^i)$:

(i) $p \in \boldsymbol{p}$ iff $P^{i-1}_{pw}(u) \in \boldsymbol{p'}$ iff $Y_{pw\overline{w}'}(u) \neq y$, by (i′) and (i″),
(ii) $q \in \boldsymbol{q}$ iff $P^{i-1}_{q\overline{w}'}(u), P^{i-1}_{\hat{p}^j \overline{w}'}(u) \in \boldsymbol{q'}$ iff $Y_{qw'\overline{w}''}(u), Y_{\hat{p}^j w'\overline{w}''}(u) = y$, for all $W' \subseteq W^i$, $w' = w|W'$, $\overline{W}'' \subseteq \overline{W}'$, $\overline{w}'' = \overline{w}'|\overline{W}''$, and $j \in \{1, \ldots, \min(i, k-1)\}$, by (ii′) and (ii″).

That is, $(\boldsymbol{p}, \boldsymbol{q}) \in R^i$ iff some $\overline{w} \in D(\overline{W})$, where $\overline{W} = W^1 \cup \cdots \cup W^i$, exists such that for all $p, q \in D(P^i)$:

(i) $p \in \boldsymbol{p}$ iff $Y_{p\overline{w}}(u) \neq y$,
(ii) $q \in \boldsymbol{q}$ iff $Y_{q\overline{w}'}(u), Y_{\hat{p}^j \overline{w}'}(u) = y$ for all $\overline{W}' \subseteq \overline{W}$, $\overline{w}' = \overline{w}|\overline{W}'$, and $j \in \{1, \ldots, \min(i, k-1)\}$. □

Proof of Theorem 6.3. By Theorem 6.2, $X = x$ is a weak cause of $Y = y$ under u in M iff (α) $X(u) = x$ and $Y(u) = y$ in M, and (β) some $\boldsymbol{p} \in R^k$ exists such that $\boldsymbol{p} \neq \emptyset$ and $x \notin \boldsymbol{p}$. It is thus sufficient to show that deciding whether (α) and (β) hold can be done in polynomial time. By Proposition 2.1, deciding whether (α) holds can be done in polynomial time. Next, we observe that $P^0, \ldots, P^k$ and $W^1, \ldots, W^k$ can be computed in time $O(\|M\|)$. By Proposition 2.1, every $\hat{p}^i$ with $i \in \{1, \ldots, k\}$ can be computed in polynomial time. We then iteratively compute every R^i with $i \in \{0, \ldots, k\}$. Clearly, R^0 can be computed in constant time, since V is domain-bounded. Observe then that the cardinality of each $D(W^i)$ is bounded by a constant, since V is domain-bounded and $G_V(M)$ is bounded. Furthermore, the size of each R^{i-1} and the cardinality of each $D(P^i)$ are both bounded by a constant, since V is domain-bounded. By Proposition 2.1, $P^{i-1}_{pw}(u)$, $P^{i-1}_{qw'}(u)$, and $P^{i-1}_{\hat{p}^j w'}(u)$ can be computed in polynomial time. Hence, every R^i can be computed by a constant number of polynomial computations, and thus in polynomial time. Hence, R^k can be computed in polynomial time. Given R^k, deciding whether (β) holds can be done in constant time. In summary, computing R^k and deciding whether (β) holds, and thus deciding whether (α) and (β) hold, can be done in polynomial time.

By Theorem 2.3, since X is a singleton, $X = x$ is an actual cause of $Y = y$ under u in M iff $X = x$ is a weak cause of $Y = y$ under u in M. So, deciding whether $X = x$ is an actual cause of $Y = y$ under u in M is also possible in polynomial time. □

Proof of Theorem 6.4. By Theorem 5.10, $X = x$ is a weak cause of $Y = y$ under u in M iff $X = x$ is a weak cause of $Y = y$ under u in $\widehat{M}^Y_X$. By Proposition 5.11, $\widehat{M}^Y_X$ is computable in polynomial time. By Theorem 6.3, given $\widehat{M}^Y_X$,

deciding whether $X = x$ is a weak cause of $Y = y$ under u in $\widehat{M}_X^Y$ can be done in polynomial time. In summary, deciding whether $X = x$ is a weak cause of $Y = y$ under u in $\widehat{M}_X^Y$ and thus in M is possible in polynomial time. □

Proof of Theorem 6.5. Recall that $X = x$ is an explanation of $Y = y$ relative to $\mathcal{C}$ iff **EX1** $Y(u) = y$ for every $u \in \mathcal{C}$, **EX2** $X = x$ is a weak cause of $Y = y$ under every $u \in \mathcal{C}$ such that $X(u) = x$, **EX3** X is minimal, and **EX4** $X(u) = x$ and $X(u') \neq x$ for some $u, u' \in \mathcal{C}$. By Proposition 2.1, checking whether **EX1** and **EX4** hold can be done in polynomial time. Clearly, **EX3** always holds, since X is a singleton. By Theorem 6.4, deciding whether $X = x$ is a weak cause of $Y = y$ under some $u \in \mathcal{C}$ in M such that $X(u) = x$ can be done in polynomial time. Thus, by Proposition 2.1, deciding whether **EX2** holds can be done in polynomial time. In summary, deciding whether **EX1**–**EX4** hold can be done in polynomial time. □

Proof of Theorem 6.6. We first compute the set $\mathcal{C}^\star$ of all $u \in \mathcal{C}$ such that either (i) $X(u) \neq x$ in M, or (ii) $X(u) = x$ and $X = x$ is a weak cause of $Y = y$ under u in M. By Proposition 2.1 and Theorem 6.4, this can be done in polynomial time. If $X = x$ is a partial explanation of $Y = y$ relative to $(\mathcal{C}, P)$ in M, then $\mathcal{C}_{X=x}^{Y=y}$ is defined, and $\mathcal{C}_{X=x}^{Y=y} = \mathcal{C}^\star$ by Proposition 2.4. Given $\mathcal{C}_{X=x}^{Y=y}$, the explanatory power $P(\mathcal{C}_{X=x}^{Y=y} \mid X = x)$ is computable in polynomial time by Proposition 2.1, if we assume as usual that P is computable in polynomial time. In summary, this shows (c).

To check partial (resp., α-partial) explanations in (a) (resp., (b)), we compute $\mathcal{C}^\star$ as above. We then check whether $\mathcal{C}_{X=x}^{Y=y}$ is defined. That is, by Proposition 2.4, we check that $X = x$ is an explanation of $Y = y$ relative to $\mathcal{C}^\star$ in M, which is possible in polynomial time by Theorem 6.5. Then, $\mathcal{C}_{X=x}^{Y=y} = \mathcal{C}^\star$ by Proposition 2.4. We finally compute $P(\mathcal{C}_{X=x}^{Y=y} \mid X = x)$ as above and check that it is positive (resp., at least α), which can clearly be done in polynomial time. In summary, this proves (a) (resp., (b)). □

Appendix C. Proofs for Section 7

Proof of Theorem 7.1. Obviously, (α) coincides with **AC1**. We now prove that (β) is equivalent to **AC2**:

AC2. Some $W \subseteq V \setminus X$ and some $\bar{x} \in D(X)$ and $w \in D(W)$ exist such that:
(a) $\neg\phi_{\bar{x}w}(u)$,
(b) $\phi_{xw'\hat{z}}(u)$ for all $W' \subseteq W$, $w' = w|W'$, $\hat{Z} \subseteq V \setminus (X \cup W)$, and $\hat{z} = \hat{Z}(u)$.

By **D6**, the variables in S^k have no parents in $G_V(M)$. Hence, every variable in S^k only depends on the variables in U, and thus we can move any $A \in S^k \setminus (W \cup X)$ into W by setting $w(A) = A(u)$. We can thus assume that $X = S^k \setminus W$ holds. Since (i) $W \subseteq V$ and $X = S^k \setminus W$ implies $W \subseteq V \setminus X$, and (ii) $X = S^k \setminus W$ implies $S^k \cup W = X \cup W$, it is thus sufficient to show that (β) is equivalent to **AC2***:

AC2***.** Some $W \subseteq V$, $\bar{x} \in D(X)$, and $w \in D(W)$ exist such that $X = S^k \setminus W$ and
(a) $\neg\phi_{\bar{x}w}(u)$,
(b) $\phi_{xw'\hat{Z}(u)}(u)$ for all $W' \subseteq W$, $w' = w|W'$, and $\hat{Z} \subseteq V \setminus (S^k \cup W)$.

We now prove that $(\star)$ for all $i \in \{0, \ldots, k\}$, $(\boldsymbol{p}, \boldsymbol{q}, F) \in R^i$ iff some $\overline{W} \subseteq T^0 \cup \cdots \cup T^i$ and $\overline{w} \in D(\overline{W})$ exist such that $F = S^i \setminus \overline{W}$ and

(i) for every $p, q \in D(F)$:
(i.1) $p \in \boldsymbol{p}$ iff $\neg\phi_{p\overline{w}}(u)$,
(i.2) $q \in \boldsymbol{q}$ iff $\phi_{[q\langle\hat{Z}(u)]\overline{w}'}(u)$ for all $\overline{W}' \subseteq \overline{W}$, $\overline{w}' = \overline{w}|\overline{W}'$, and $\hat{Z} \subseteq (T^0 \cup \cdots \cup T^i) \setminus (S^k \cup \overline{W})$.

In particular, this then implies that $(\boldsymbol{p}, \boldsymbol{q}, F) \in R^k$ iff some $\overline{W} \subseteq T^0 \cup \cdots \cup T^k = V$ and $\overline{w} \in D(\overline{W})$ exist such that $F = S^k \setminus \overline{W}$ and

(i) for every $p, q \in D(F)$:

(i.1) $p \in \boldsymbol{p}$ iff $\neg\phi_{p\overline{w}}(u)$,
(i.2) $q \in \boldsymbol{q}$ iff $\phi_{[q\langle\hat{Z}(u)]\overline{w}'}(u)$ for all $\overline{W}' \subseteq \overline{W}$, $\overline{w}' = \overline{w}|\overline{W}'$, and $\hat{Z} \subseteq (T^0 \cup \cdots \cup T^k) \setminus (S^k \cup \overline{W}) = V \setminus (S^k \cup \overline{W})$.

This then shows that **AC2*** is equivalent to (β) some $(\boldsymbol{p}, \boldsymbol{q}, X) \in R^k$ exists such that $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$: ($\Leftarrow$) Suppose first that (β) holds. Hence, some $\overline{W} \subseteq V$ and some $\overline{w} \in D(\overline{W})$ exist such that $X = S^k \setminus \overline{W}$ and (a) $\neg\phi_{p\overline{w}}(u)$ for some $p \in \boldsymbol{p} \neq \emptyset$, and (b) $\phi_{[q\langle\hat{Z}(u)]\overline{w}'}(u)$ for $q = x \in \boldsymbol{q}$ and all $\overline{W}' \subseteq \overline{W}$, $\overline{w}' = \overline{w}|\overline{W}'$, and $\hat{Z} \subseteq V \setminus (S^k \cup \overline{W})$. That is, **AC2*** holds. ($\Rightarrow$) Conversely, suppose now that **AC2*** holds. Let $(\boldsymbol{p}, \boldsymbol{q}, X)$ be defined by (i), using $W \subseteq V$ and $w \in D(W)$ from **AC2*** as $\overline{W} \subseteq V$ and $\overline{w} \in D(\overline{W})$, respectively. Then, $(\boldsymbol{p}, \boldsymbol{q}, X) \in R^k$, $\boldsymbol{p} \neq \emptyset$, and $x \in \boldsymbol{q}$. That is, (β) holds.

We give a proof of ($\star$) by induction on $i \in \{0, \ldots, k\}$:

Basis: Recall that $(\boldsymbol{p}, \boldsymbol{q}, F) \in R^0$ iff some $W \subseteq T^0$ and $w \in D(W)$ exist such that $F = S^0 \setminus W$ and

(i) for every $p, q \in D(F)$:
(i.1) $p \in \boldsymbol{p}$ iff $\neg\phi_{pw}(u)$,
(i.2) $q \in \boldsymbol{q}$ iff $\phi_{[q\langle\hat{Z}(u)]w'}(u)$ for all $W' \subseteq W$, $w' = w|W'$, and $\hat{Z} \subseteq T^0 \setminus (S^k \cup W)$.

Induction: Recall that $(\boldsymbol{p}, \boldsymbol{q}, F) \in R^i$ iff some $W \subseteq T^i$, $w \in D(W)$, and $(\boldsymbol{p'}, \boldsymbol{q'}, F') \in R^{i-1}$ exist such that $F = S^i \setminus W$ and

(i′) for every $p, q \in D(F)$:
(i.1′) $p \in \boldsymbol{p}$ iff ${F'}_{pw}(u) \in \boldsymbol{p'}$,
(i.2′) $q \in \boldsymbol{q}$ iff ${F'}_{[q\langle\hat{Z}(u)]w'}(u) \in \boldsymbol{q'}$ for all $W' \subseteq W$, $w' = w|W'$, and $\hat{Z} \subseteq T^i \setminus (S^k \cup W)$.

The induction hypothesis says that $(\boldsymbol{p'}, \boldsymbol{q'}, F') \in R^{i-1}$ iff some $\overline{W}' \subseteq T^0 \cup \cdots \cup T^{i-1}$ and $\overline{w}' \in D(\overline{W}')$ exist such that $F' = S^{i-1} \setminus \overline{W}'$ and

(i″) for every $p', q' \in D(F')$:
(i.1″) $p' \in \boldsymbol{p'}$ iff $\neg\phi_{p'\overline{w}'}(u)$,
(i.2″) $q' \in \boldsymbol{q'}$ iff $\phi_{[q'\langle\hat{Z}'(u)]\overline{w}''}(u)$ for all $\overline{W}'' \subseteq \overline{W}'$, $\overline{w}'' = \overline{w}'|\overline{W}''$, and $\hat{Z}' \subseteq (T^0 \cup \cdots \cup T^{i-1}) \setminus (S^k \cup \overline{W}')$.

It thus follows that $(\boldsymbol{p}, \boldsymbol{q}, F) \in R^i$ iff some $\overline{W}' \subseteq T^0 \cup \cdots \cup T^{i-1}$, $W \subseteq T^i$, $\overline{w}' \in D(\overline{W}')$, and $w \in D(W)$ exist such that $F = S^i \setminus W$ and

(i‴) for $F' = S^{i-1} \setminus \overline{W}'$ and every $p, q \in D(F)$:
(i.1‴) $p \in \boldsymbol{p}$ iff $\neg\phi_{p'\overline{w}'}(u)$, where $p' = {F'}_{pw}(u)$, by (i.1′) and (i.1″),
(i.2‴) $q \in \boldsymbol{q}$ iff $\phi_{[q'\langle\hat{Z}'(u)]\overline{w}''}(u)$, where $q' = {F'}_{[q\langle\hat{Z}(u)]w'}(u)$, for all $\overline{W}'' \subseteq \overline{W}'$, $\overline{w}'' = \overline{w}'|\overline{W}''$, and $\hat{Z}' \subseteq (T^0 \cup \cdots \cup T^{i-1}) \setminus (S^k \cup \overline{W}')$, and all $W' \subseteq W$, $w' = w|W'$, and $\hat{Z} \subseteq T^i \setminus (S^k \cup W)$, by (i.2′) and (i.2″).

By **D4–D6** in the definition of a decomposition, setting some of the T^i-variables as W- or $\hat{Z}$-variables in (i.1‴) and (i.2‴) does not influence the values of the variables in $S^i \setminus (W \cup \hat{Z})$. Thus, ${F'}_{pw}(u) = {F'}_{pw\overline{w}'}(u)$, and so $\neg\phi_{p'\overline{w}'}(u) = \neg\phi_{pw\overline{w}'}(u)$. Furthermore, it holds $A_{[q\langle\hat{Z}(u)]w'}(u) = A_{[q\langle\hat{Z}(u)]w'\hat{Z}'(u)\overline{w}''}(u)$ for all $A \in F' \setminus \hat{Z}'$, and thus $\phi_{[q'\langle\hat{Z}'(u)]\overline{w}''}(u)$, where $q' = {F'}_{[q\langle\hat{Z}(u)]w'}(u)$, is equivalent to $\phi_{[q\langle\hat{Z}(u)]w'\hat{Z}'(u)\overline{w}''}(u) = \phi_{[q\langle\hat{Z}(u)\hat{Z}'(u)]w'\overline{w}''}(u)$. Hence, it follows that $(\boldsymbol{p}, \boldsymbol{q}, F) \in R^i$ iff some $\overline{W}' \subseteq T^0 \cup \cdots \cup T^{i-1}$, $W \subseteq T^i$, $\overline{w}' \in D(\overline{W}')$, and $w \in D(W)$ exist such that $F = S^i \setminus W$ and

(i) for every $p, q \in D(F)$:
(i.1) $p \in \boldsymbol{p}$ iff $\neg\phi_{pw\overline{w}'}(u)$,
(i.2) $q \in \boldsymbol{q}$ iff $\phi_{[q\langle\hat{Z}(u)]w'\overline{w}''}(u)$ for all $\overline{W}'' \subseteq \overline{W}'$, $\overline{w}'' = \overline{w}'|\overline{W}''$, $W' \subseteq W$, $w' = w|W'$, and $\hat{Z} \subseteq (T^0 \cup \cdots \cup T^i) \setminus (S^k \cup W \cup \overline{W}')$.

That is, it holds that $(\boldsymbol{p}, \boldsymbol{q}, F) \in R^i$ iff some $\overline{W} \subseteq T^0 \cup \cdots \cup T^i$ and $\overline{w} \in D(\overline{W})$ exist such that $F = S^i \setminus \overline{W}$ and

(i) for every $p, q \in D(F)$:
  (i.1) $p \in \boldsymbol{p}$ iff $\neg\phi_{p\overline{w}}(u)$,
  (i.2) $q \in \boldsymbol{q}$ iff $\phi_{[q\langle\hat{Z}(u)]\overline{w}'}(u)$ for all $\overline{W}' \subseteq \overline{W}$, $\overline{w}' = \overline{w}|\overline{W}'$, and $\hat{Z} \subseteq (T^0 \cup \cdots \cup T^i) \setminus (S^k \cup \overline{W})$. □

Proof of Theorem 7.2. Observe first that X is bounded by a constant, since there exists a width-bounded decomposition of $G_V(M)$ relative to X and ϕ. Hence, it is sufficient to prove the statement of the theorem for the notion of weak cause. Let $\mathcal{D} = ((T^0, S^0), \ldots, (T^k, S^k))$. By Theorem 7.1, $X = x$ is a weak cause of ϕ under u in M iff (α) $X(u) = x$ and $\phi(u)$ in M, and (β) some $(\boldsymbol{p}, \boldsymbol{q}, X) \in R^k$ exists such that $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$, where R^k is computed using the decomposition $\mathcal{D}$ of $G_V(M)$ relative to X and ϕ. By Proposition 2.2, deciding whether (α) holds can be done in polynomial time. Since V is domain-bounded and $\mathcal{D}$ is width-bounded, it follows that R^0 can be computed in polynomial time, and that each R^i, $i \in \{1, \ldots, k\}$, can be computed in polynomial time from R^{i-1}. Hence, R^k can be computed in polynomial time. Since V is domain-bounded and $\mathcal{D}$ is width-bounded, it then follows that, given R^k, checking (β) can be done in constant time. In summary, deciding whether (β) holds can also be done in polynomial time. □

Proof of Theorem 7.3. We first prove the statement of the theorem for the notion of weak cause. Let $X'' = X' \cap R^{\phi}_X(M)$ and $x'' = x'|X''$. By Theorem 5.7, $X' = x'$ is a weak cause of ϕ under u in M iff (i) $(X' \setminus X'')(u) = x'|(X' \setminus X'')$ in M, and (ii) $X'' = x''$ is a weak cause of ϕ under u in M^{ϕ}_X. By Proposition 5.8, $R^{\phi}_X(M)$ can be computed in linear time, and thus $X' \setminus X'' = X' \setminus R^{\phi}_X(M)$ can be computed in linear time. By Proposition 2.1, given $X' \setminus X''$, checking (i) can be done in polynomial time. In summary, deciding whether (i) holds can be done in polynomial time. By Proposition 5.8, M^{ϕ}_X can be computed in polynomial time. By Theorem 7.2, given M^{ϕ}_X, checking (ii) can be done in polynomial time. In summary, deciding whether (ii) holds can be done in polynomial time. Tractability also holds for actual causes, since there exists a width-bounded decomposition of $G_V(M^{\phi}_X)$ relative to $X' \cap R^{\phi}_X(M)$ and ϕ, and thus $X' \cap R^{\phi}_X(M)$ is bounded by a constant. □

Proof of Theorem 7.4. We first prove the statement of the theorem for the notion of weak cause. Let $X' = X \cap \widehat{R}^{\phi}_X(M)$ and $x' = x|X'$. By Theorem 5.10, $X = x$ is a weak cause of ϕ under u in M iff (i) $(X \setminus X')(u) = x|(X \setminus X')$ in M, and (ii) $X' = x'$ is a weak cause of ϕ under u in $\widehat{M}^{\phi}_X$. By Proposition 5.11, $\widehat{R}^{\phi}_X(M)$ can be computed in linear time, and thus $X \setminus X' = X \setminus \widehat{R}^{\phi}_X(M)$ can be computed in linear time. By Proposition 2.1, given $X \setminus X'$, checking (i) can be done in polynomial time. In summary, deciding whether (i) holds can be done in polynomial time. By Proposition 5.11, $\widehat{M}^{\phi}_X$ can be computed in polynomial time. By Theorem 7.2, given $\widehat{M}^{\phi}_X$, checking (ii) can be done in polynomial time. In summary, deciding whether (ii) holds can be done in polynomial time. Tractability also holds for actual causes, since there exists a width-bounded decomposition of $G_V(\widehat{M}^{\phi}_X)$ relative to $X \cap \widehat{R}^{\phi}_X(M)$ and ϕ, and thus $X \cap \widehat{R}^{\phi}_X(M)$ is bounded by a constant. □

Proof of Theorem 7.5. Since $\mathcal{D}$ (resp., $\mathcal{D}_X$) for (a) (resp., (b)) is width-bounded, it follows that $|X|$ is bounded by a constant. Hence, it is sufficient to prove the statement of the theorem for the notion of weak cause. Furthermore, if $X' = x'$ is a weak cause of ϕ under u in M, then $X'(u) = x'$ in M. Hence, it is sufficient to show that for every $X' \subseteq X$ and $x' \in D(X)$, where $x' = X'(u)$ in M, deciding whether $X' = x'$ is a weak cause of ϕ under u in M can be done in polynomial time. Observe then for (a) (resp., (b)) that $\mathcal{D}$ (resp., $\mathcal{D}_{X'}$) is also a decomposition of $G_V(M^{\phi}_X)$ (resp., $G_V(\widehat{M}^{\phi}_{X'})$) relative to $X' \cap R^{\phi}_X(M)$ (resp., $X' \cap \widehat{R}^{\phi}_{X'}(M)$) and ϕ. By Theorem 7.3 (resp., 7.4) for (a) (resp., (b)), it thus follows that deciding whether $X' = x'$ is a weak cause of ϕ under u in M can be done in polynomial time.

In case (a), by exploiting the independence of $R^{\phi}_X(M)$ from X, we can proceed as follows, avoiding multiple computations of the set R^k. First, check that $\phi(u)$ holds and compute R^k for $\mathcal{D}$ and $X'' = X \cap R^{\phi}_X(M)$. Then, for each subset $X' \subseteq X''$ such that some triple $(\boldsymbol{p}, \boldsymbol{q}, X')$ exists in R^k such that $\boldsymbol{p} \neq \emptyset$ and $x' \in \boldsymbol{q}$, where $x' = X'(u)$ in M, we have that $X' = x'$ is a weak cause of ϕ under u in M. Extending each such X' by an arbitrary subset Z of variables

from $X \setminus X''$, we obtain that $X'Z = x'z$, where $z = Z(u)$ in M, is also a weak cause of ϕ under u. In this way, all weak causes $X' = x'$ for ϕ under u in M where $X' \subseteq X$ can be computed. □

Proof of Theorem 7.6. Recall that $X = x$ is an explanation of ϕ relative to $\mathcal{C}$ iff **EX1** $\phi(u)$ for every $u \in \mathcal{C}$, **EX2** $X = x$ is a weak cause of ϕ under every $u \in \mathcal{C}$ such that $X(u) = x$, **EX3** X is minimal, that is, for every $X' \subset X$, some $u \in \mathcal{C}$ exists such that (1) $X'(u) = x|X'$ and (2) $X' = x|X'$ is not a weak cause of ϕ under u, and **EX4** $X(u) = x$ and $X(u') \neq x$ for some $u, u' \in \mathcal{C}$. By Proposition 2.2, checking whether **EX1** and **EX4** hold can be done in polynomial time. By Theorem 7.3 (resp., 7.4) for (a) (resp., (b)), deciding whether $X = x$ is a weak cause of ϕ under some $u \in \mathcal{C}$ such that $X(u) = x$ can be done in polynomial time. Thus, by Proposition 2.1, deciding whether **EX2** holds can be done in polynomial time. We finally show that checking **EX3** is possible in polynomial time. For (a) (resp., (b)), observe that $\mathcal{D}$ (resp., $\mathcal{D}_{X'}$) is also a decomposition of $G_V(M_X^\phi)$ (resp., $G_V(\widehat{M}_{X'}^\phi)$) relative to $X' \cap R_X^\phi(M)$ (resp., $X' \cap \widehat{R}_{X'}^\phi(M)$) and ϕ, for each $X' \subset X$. Since $\mathcal{D}$ (resp., $\mathcal{D}_X$) for (a) (resp., (b)) is width-bounded, it follows that $|X|$ is bounded by a constant. By Proposition 2.1 and Theorem 7.3 (resp., 7.4) for (a) (resp., (b)), deciding whether (1) $X'(u) = x|X'$ and (2) $X' = x|X'$ is not a weak cause of ϕ under some $u \in \mathcal{C}$ can be done in polynomial time, for every $X' \subset X$. Hence, deciding whether **EX3** holds can be done in polynomial time. In summary, deciding whether **EX1** to **EX4** hold can be done in polynomial time. □

Proof of Theorem 7.7. We first compute the set $\mathcal{C}^\star$ of all $u \in \mathcal{C}$ such that either (i) $X(u) \neq x$ in M, or (ii) $X(u) = x$ and $X = x$ is a weak cause of ϕ under u in M. By Proposition 2.1 and Theorem 7.3 (resp., 7.4) for (a) (resp., (b)), this can be done in polynomial time. If $X = x$ is a partial explanation of ϕ relative to $(\mathcal{C}, P)$ in M, then $\mathcal{C}_{X=x}^\phi$ is defined, and $\mathcal{C}_{X=x}^\phi = \mathcal{C}^\star$ by Proposition 2.4. Given $\mathcal{C}_{X=x}^\phi$, the explanatory power $P(\mathcal{C}_{X=x}^\phi \mid X = x)$ is computable in polynomial time by Proposition 2.1, if P is computable in polynomial time, as usual. In summary, this shows (3).

To check partial (resp., α-partial) explanations in (1) (resp., (2)), we compute $\mathcal{C}^\star$ as above. We then check that $\mathcal{C}_{X=x}^\phi$ is defined. That is, by Proposition 2.4, we check that $X = x$ is an explanation of ϕ relative to $\mathcal{C}^\star$ in M, which is possible in polynomial time by Theorem 7.6. Then, $\mathcal{C}_{X=x}^\phi = \mathcal{C}^\star$ by Proposition 2.4. We finally compute $P(\mathcal{C}_{X=x}^\phi \mid X = x)$ as above and check that it is positive (resp., at least α), which can be done in polynomial time. In summary, this proves (1) (resp., (2)). □

Proof of Theorem 7.8. Observe that the set of all $X' = x'$ such that $X' \subseteq X$ and $x' \in D(X')$ is bounded by a constant, since V is domain-bounded, and $\mathcal{D}$ (resp., $\mathcal{D}_X$) for (a) (resp., (b)) is width-bounded, and thus $|X|$ is bounded by a constant. Hence, it is sufficient to show that for every $X' \subseteq X$ and $x' \in D(X')$, deciding whether $X' = x'$ is an explanation of ϕ relative to $\mathcal{C}$ in M is possible in polynomial time. This can be done in a similar way as the proof of Theorem 7.6. □

Proof of Theorem 7.9. As argued in the proof of Theorem 7.8, the set of all $X' = x'$ such that $X' \subseteq X$ and $x' \in D(X')$ is bounded by a constant. Hence, it is sufficient to show that for every $X' \subseteq X$ and $x' \in D(X')$, deciding whether $X' = x'$ is a partial (resp., an α-partial) explanation of ϕ relative to $(\mathcal{C}, P)$ in M is possible in polynomial time. This can be done as in the proof of Theorem 7.7 (1) (resp., (2)). □

Proof of Theorem 7.10. We generalize the proof of Theorem 7.1. We show that some $(\boldsymbol{p}, \boldsymbol{q}, X, l) \in R^k$ exists with $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$ iff **AC2l** holds:

AC2l. Some $W \subseteq V \setminus X$ and some $\bar{x} \in D(X)$ and $w \in D(W)$ exist such that:
(a) $\neg\phi_{\bar{x}w}(u)$,
(b) $\phi_{xw'\hat{Z}(u)}(u)$ for all $W' \subseteq W$, $w' = w|W'$, and $\hat{Z} \subseteq V \setminus (X \cup W)$,
(c) $\mathrm{diff}(w, W(u)) = l$.

As in the proof of Theorem 7.1, by moving any $A \in S^k \setminus (W \cup X)$ into W by setting $w(A) = A(u)$ (which does not influence $\mathrm{diff}(w, W(u))$), it is sufficient to show that some $(\boldsymbol{p}, \boldsymbol{q}, X, l) \in R^k$ exists with $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$ iff **AC2l*** holds:

AC2l*. Some $W \subseteq V$, $\bar{x} \in D(X)$, and $w \in D(W)$ exist such that $X = S^k \setminus W$ and
(a) $\neg\phi_{\bar{x}w}(u)$,
(b) $\phi_{xw'\hat{Z}(u)}(u)$ for all $W' \subseteq W$, $w' = w|W'$, and $\hat{Z} \subseteq V \setminus (S^k \cup W)$,
(c) $\mathrm{diff}(w, W(u)) = l$.

This can be done in a similar way as showing that (β) is equivalent to **AC2*** in the proof of Theorem 7.1, where we use the following result ($\star\star$) instead of ($\star$), which can be proved by induction on $i \in \{0, \ldots, k\}$ (in a similar way as ($\star$)): ($\star\star$) For all $i \in \{0, \ldots, k\}$, it holds that $(\boldsymbol{p}, \boldsymbol{q}, F, l) \in R^i$ iff some $\overline{W} \subseteq T^0 \cup \cdots \cup T^i$ and $\overline{w} \in D(\overline{W})$ exist such that $F = S^i \setminus \overline{W}$, $\mathrm{diff}(\overline{w}, \overline{W}(u)) = l$, and

(i) for every $p, q \in D(F)$:
(i.1) $p \in \boldsymbol{p}$ iff $\neg\phi_{p\overline{w}}(u)$,
(i.2) $q \in \boldsymbol{q}$ iff $\phi_{[q\langle\hat{Z}(u)]\overline{w}'}(u)$ for all $\overline{W}' \subseteq \overline{W}$, $\overline{w}' = \overline{w}|\overline{W}'$, and $\hat{Z} \subseteq (T^0 \cup \cdots \cup T^i) \setminus (S^k \cup \overline{W})$. □

Proof of Theorem 7.11. We first decide if ($\star$) $X = x$ is an actual cause of ϕ under u in M, which can be done in polynomial time by Theorem 7.2. If ($\star$) does not hold, then $\mathrm{dr}((M, u), X = x, \phi) = 0$. Otherwise, $\mathrm{dr}((M, u), X = x, \phi) = 1/(l^\star + 1)$, where $l^\star$ is the minimal l for which some $W \subseteq V \setminus X$, $\bar{x} \in D(X)$, and $w \in D(W)$ exist such that **AC2**(a) and (b) hold and $\mathrm{diff}(w, W(u)) = l$. By Theorem 7.10, $l^\star$ is the minimal l for which some $(\boldsymbol{p}, \boldsymbol{q}, X, l) \in R^k$ exists such that $\boldsymbol{p} \neq \emptyset$ and $x \in \boldsymbol{q}$. Since V is domain-bounded and $\mathcal{D}$ is width-bounded, R^0 can be computed in polynomial time, and each R^i, $i \in \{1, \ldots, k\}$, can be computed in polynomial time from R^{i-1}. Thus, R^k can be computed in polynomial time. Since V is domain-bounded and $\mathcal{D}$ is width-bounded, $l^\star$ can be computed in polynomial time from R^k. In summary, $l^\star$ and thus $\mathrm{dr}((M, u), X = x, \phi) = 1/(l^\star + 1)$ can be computed in polynomial time. □

Proof of Theorem 7.12. By Theorem 7.11, every $\mathrm{dr}((M, u), X = x, \phi)$, $(M, u) \in \mathcal{K}$, can be computed in polynomial time. Assuming that P can be computed in polynomial time, also $\mathrm{db}(\mathcal{K}, P, X \leftarrow x, \phi)$ can be computed in polynomial time. □

Appendix D. Proofs for Section 8

Proof of Proposition 8.1. Let $(S^0, \ldots, S^k)$ be an arbitrary layering of $G_V(M)$ w.r.t. X and ϕ. We now show that $((T^0, S^0), \ldots, (T^k, S^k))$, where $T^0 = S^0, \ldots, T^k = S^k$, is a decomposition of $G_V(M)$ w.r.t. X and ϕ, that is, that **D1**–**D6** hold. Trivially, **D1** and **D2** hold. Moreover, **L2** implies **D3**, and **L1** implies **D4**–**D6**. □

Proof of Proposition 8.2. Assume that $\mathcal{L} = (S^0, \ldots, S^k)$ is an arbitrary layering of $G_V(M)$ relative to X and ϕ. By **L2**, every $A \in V(\phi) \cap V$ belongs to S^0, and at least one such variable exists. By **L2** and since $G_V(M)$ is connected relative to X and ϕ, every variable $A \in X$ belongs to S^k, and at least on such variable exists, where k is given via a path P from a variable $B \in V(\phi)$ to a variable in X (in the undirected graph for $G_V(M)$) as the number of arrows in $G_V(M)$ that go against the direction of P minus the number of arrows in $G_V(M)$ that go in the same direction as P. Indeed, if we move from B to A (against the direction of P), any step backwards toward S^i must be compensated later with a step forward. By **L1** and since $G_V(M)$ is connected relative to X and ϕ, for every $i \in \{0, \ldots, k\}$, the set S^i is the set of all $A \in V$ that are reachable from some $B \in X \cup V(\phi)$ on a path P (in the undirected graph for $G_V(M)$) such that i is the number of arrows in $G_V(M)$ that go against the direction of P minus the number of arrows in $G_V(M)$ that go in the same direction as P plus j with $B \in S^j$. That is, the layering $\mathcal{L}$ is unique. □

Proof of Proposition 8.3. In Step (1), we initialize $\lambda(A)$ to undefined for all $A \in V \setminus V(\phi)$. In Step (2), every variable occurring in ϕ is put into S^0, in order to satisfy one part of **L2**. In Steps (3)–(13), since $G_V(M)$ is connected, all the other variables are put into some S^j such that **L1** is satisfied. Step (3) takes care of the special case in which variables from ϕ belong to X, where then only a trivial layered decomposition is possible. Steps (6) and (11) catch cases in which no layering mapping as desired exists, and then *Nil* is returned. Notice that the for-loop in Step (9) is executed at most once. Finally, we check in Steps (14) and (15) that $X \subseteq S^k$, where k is the maximal index j of some S^j, and thus whether the other part of **L2** is also satisfied. If so, then we return the computed layering λ; otherwise, we return *Nil*. □

Proof of Proposition 8.4. By Proposition 8.2, if a layering of $G_V(M)$ relative to X and ϕ exists, then it is unique. By Proposition 8.3, Algorithm LAYERING returns the unique layering $\mathcal{L}$ of $G_V(M)$ relative to X and ϕ, if it exists, and *Nil*, otherwise. Observe then that Steps (1)–(3) of LAYERING take $O(|V| + |V(\phi)|)$ time, Steps (4)–(13) take $O(|E| + |X|)$ time, and Step (14) is feasible in $O(|V|)$ time (using an auxiliary variable for the maximum of λ, even in constant time). Hence, LAYERING can be implemented to run in $O(|V| + |V(\phi)| + |E|)$ time, i.e., in $O(\|G_V(M)\| + |V(\phi)|)$ time. Given that $G_V(M)$ is layered, deciding whether $\mathcal{L}$ is width-bounded by some integer $l \geqslant 0$ can be done in time in $O(|V|)$. □

Appendix E. Proofs for Section 9

Proof of Theorem 9.1. The proof is nearly identical to the proof of Theorem 5.1 (resp., 5.2), except that **AC2** is now replaced by **AC2″** (for the notion of weak cause in extended causal models). In the "⇒"-part, we use that $\bar{x}''w$ is allowable if $\bar{x}w$ is allowable, while in the "⇐"-part, we use that $\bar{x}''\bar{x}_0w'$ is allowable if $\bar{x}''w$ is allowable, which follows from the assumption that M is closed relative to X''. □

Proof of Theorem 9.2. (a) Let $V' = R_X^\phi(M)$ and $M' = M_X^\phi$. Assume M is closed. Let $Y \subseteq V'$, let y be an allowable setting for Y in M', and let $u \in D(U)$. Then, y is an allowable setting for Y in M, and $(V' \setminus Y)_y(u)$ has the same value in M and M'. Since M is closed, $y \cup (V \setminus Y)_y(u)$ is an allowable setting for Y in M, and thus $y \cup (V' \setminus Y)_y(u)$ is an allowable setting for Y in M'. Hence, M' is closed.

(b) Let $V' = \widehat{R}_X^\phi(M)$ and $M' = \widehat{M}_X^\phi$. Suppose M is closed relative to X'. Let $Y \subseteq V'$ with $X' \subseteq Y$, let y be an allowable setting for Y in M', and let $u \in D(U)$. Then, y is an allowable setting for Y in M, and $(V' \setminus Y)_y(u)$ has the same value in M and M'. Since M is closed relative to X', it follows that $y \cup (V \setminus Y)_y(u)$ is an allowable setting for Y in M, and thus $y \cup (V' \setminus Y)_y(u)$ is an allowable setting for Y in M'. This shows that M' is closed relative to X'. □

Proof of Theorem 9.3. The proof is similar to the proof of Theorem 5.5. We only replace **AC2** by **AC2″** (for the notion of weak cause in extended causal models). In the "⇒"-part, we then use that $\bar{x}''\overline{w}$ is allowable in M_X^ϕ if $\bar{x}'w$ is allowable in M, while in the "⇐"-part, we use that $\bar{x}'w$ is allowable in M if $\bar{x}''w$ is allowable in M_X^ϕ, which follows from M being closed relative to X''. □

Proof of Theorem 9.4. The proof is nearly identical to the proof of Theorem 7.1, except that **AC2** is now replaced by **AC2″** (for the notion of weak cause in extended causal models), the relations R^i for the notion of weak cause are replaced by the relations R^i for the notion of weak cause in extended causal models, and $(\star')$ is replaced by the following statement $(\star'')$: for all $i \in \{0, \ldots, k\}$, it holds that $(\boldsymbol{p}, \boldsymbol{q}, F) \in R^i$ iff some $\overline{W} \subseteq T^0 \cup \cdots \cup T^i$ and $\overline{w} \in D(\overline{W})$ exist such that $F = S^i \setminus \overline{W}$ and (i) for every $p, q \in D(F)$: (i.1) $p \in \boldsymbol{p}$ iff $\neg\phi_{p\overline{w}}(u)$ and $p\overline{w}|(X \cup W)$ is allowable, and (i.2) $q \in \boldsymbol{q}$ iff $\phi_{[q\langle\hat{Z}(u)\rangle]\overline{w}'}(u)$ for all $\hat{Z} \subseteq (T^0 \cup \cdots \cup T^i) \setminus (S^k \cup \overline{W})$, $\overline{W}' \subseteq \overline{W}$, and $\overline{w}' = \overline{w}|\overline{W}'$. Observe that in the step from **AC2″** to **(AC2 ″)***, we then use the assumption that M is closed relative to X. Moreover, in the induction step, we use the property **D7** of decompositions in extended causal models. □

References

[1] A. Balke, J. Pearl, Probabilistic evaluation of counterfactual queries, in: Proceedings AAAI-1994, AAAI Press, 1994, pp. 230–237.

[2] U. Chajewska, J.Y. Halpern, Defining explanation in probabilistic systems, in: Proceedings UAI-1997, Morgan Kaufmann, San Mateo, CA, 1997, pp. 62–71.

[3] H. Chockler, J.Y. Halpern, Responsibility and blame: A structural-model approach, in: Proceedings IJCAI-2003, Morgan Kaufmann, San Mateo, CA, 2003, pp. 147–153. Extended version in J. Artificial Intelligence Res. 22 (2004) 93–115.

[4] H. Chockler, J.Y. Halpern, O. Kupferman, What causes a system to satisfy a specification?, in: CoRR, 2003.

[5] E. Dantsin, T. Eiter, G. Gottlob, A. Voronkov, Complexity and expressive power of logic programming, ACM Comput. Surv. 33 (3) (2001) 374–425.

[6] T. Eiter, T. Lukasiewicz, Complexity results for structure-based causality, in: Proceedings IJCAI-2001, Morgan Kaufmann, San Mateo, CA, 2001, pp. 35–40. Extended version in Artificial Intelligence 142 (1) (2002) 53–89.

[7] T. Eiter, T. Lukasiewicz, Complexity results for explanations in the structural-model approach, in: Proceedings KR-2002, Morgan Kaufmann, San Mateo, CA, 2002, pp. 49–60. Extended version in Artificial Intelligence 154 (1–2) (2004) 145–198.

[8] T. Eiter, T. Lukasiewicz, Causes and explanations in the structural-model approach: Tractable cases, in: Proceedings UAI-2002, Morgan Kaufmann, San Mateo, CA, 2002, pp. 146–153. Extended Report INFSYS RR-1843-02-03, Institut für Informationssysteme, TU Wien, March 2002; September 2005.
[9] A. Finzi, T. Lukasiewicz, Structure-based causes and explanations in the independent choice logic, in: Proceedings UAI-2003, Morgan Kaufmann, San Mateo, CA, 2003, pp. 225–232.
[10] D. Galles, J. Pearl, Axioms of causal relevance, Artificial Intelligence 97 (1–2) (1997) 9–43.
[11] P. Gärdenfors, Knowledge in Flux, MIT Press, Cambridge, MA, 1988.
[12] H. Geffner, Causal theories for nonmonotonic reasoning, in: Proceedings AAAI-1990, AAAI Press/MIT Press, 1990, pp. 524–530.
[13] H. Geffner, Default Reasoning: Causal and Conditional Theories, MIT Press, Cambridge, MA, 1992.
[14] J.Y. Halpern, Axiomatizing causal reasoning, J. Artificial Intelligence Res. 12 (2000) 317–337.
[15] J.Y. Halpern, J. Pearl, Causes and explanations: A structural-model approach, Technical Report R-266, UCLA Cognitive Systems Lab, 2000.
[16] J.Y. Halpern, J. Pearl, Causes and explanations: A structural-model approach—Part I: Causes, in: Proceedings UAI-2001, Morgan Kaufmann, San Mateo, CA, 2001, pp. 194–202.
[17] J.Y. Halpern, J. Pearl, Causes and explanations: A structural-model approach—Part I: Causes, British J. Philos. Sci. 56 (2005) 843–887.
[18] J.Y. Halpern, J. Pearl, Causes and explanations: A structural-model approach—Part II: Explanations, in: Proceedings IJCAI-2001, Morgan Kaufmann, San Mateo, CA, 2001, pp. 27–34. Extended version in British J. Philos. Sci. 56 (2005) 889–911.
[19] C.G. Hempel, Aspects of Scientific Explanation, Free Press, 1965.
[20] M. Henrion, M.J. Druzdzel, Qualitative propagation and scenario-based approaches to explanation of probabilistic reasoning, in: Uncertainty in Artificial Intelligence, vol. 6, Elsevier Science, Amsterdam, 1990, pp. 17–32.
[21] M. Hopkins, Strategies for determining causes of reported events, in: Proceedings AAAI-2002, AAAI Press, 2002, pp. 546–552.
[22] M. Hopkins, LAYERWIDTH: Analysis of a new metric for directed acyclic graphs, in: Proceedings UAI-2003, Morgan Kaufmann, San Mateo, CA, 2003, pp. 321–328.
[23] M. Hopkins, J. Pearl, Causality and counterfactuals in the situation calculus, Technical Report R-301, UCLA Cognitive Systems Laboratory, 2002.
[24] M. Hopkins, J. Pearl, Clarifying the usage of structural models for commonsense causal reasoning, in: Proceedings AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning, AAAI Press, 2003, pp. 83–89.
[25] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, F. Scarcello, The DLV System for knowledge representation and reasoning, ACM Trans. Comput. Logic (TOCL), in press.
[26] V. Lifschitz, On the logic of causal explanation, Artificial Intelligence 96 (2) (1997) 451–465.
[27] N. McCain, H. Turner, Causal theories of action and change, in: Proceedings AAAI-1997, AAAI Press/MIT Press, 1997, pp. 460–465.
[28] J.D. Park, Causes and explanations revisited, in: Proceedings IJCAI-2003, Morgan Kaufmann, San Mateo, CA, 2003, pp. 154–162.
[29] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, CA, 1988.
[30] J. Pearl, Reasoning with cause and effect, in: Proceedings IJCAI-1999, Morgan Kaufmann, San Mateo, CA, 1999, pp. 1437–1449.
[31] J. Pearl, Causality: Models, Reasoning, and Inference, Cambridge University Press, Cambridge, 2000.
[32] D. Poole, The independent choice logic for modelling multiple agents under uncertainty, Artificial Intelligence 94 (1–2) (1997) 7–56.
[33] D. Poole, Decision theory, the situation calculus and conditional plans, Electron. Trans. Artificial Intelligence 2 (1–2) (1998) 105–158.
[34] W.C. Salmon, Four Decades of Scientific Explanation, University of Minnesota Press, 1989.
[35] S.E. Shimony, Explanation, irrelevance, and statistical independence, in: Proceedings AAAI-1991, AAAI Press/MIT Press, 1991, pp. 482–487.
[36] H. Turner, A logic of universal causation, Artificial Intelligence 113 (1–2) (1999) 87–123.

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Artificial Intelligence 170 (2006) 581–606
Artificial Intelligence
www.elsevier.com/locate/artint

Linguistic quantifiers modeled by Sugeno integrals

Mingsheng Ying [1]

State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Received 1 December 2004; received in revised form 3 February 2006; accepted 15 February 2006

Available online 20 March 2006

Abstract

Since quantifiers have the ability of summarizing the properties of a class of objects without enumerating them, linguistic quantification is a very important topic in the field of high level knowledge representation and reasoning. This paper introduces a new framework for modeling quantifiers in natural languages in which each linguistic quantifier is represented by a family of fuzzy measures, and the truth value of a quantified proposition is evaluated by using Sugeno's integral. This framework allows us to have some elegant logical properties of linguistic quantifiers. We compare carefully our new model of quantification and other approaches to linguistic quantifiers. A set of criteria for linguistic quantification was proposed in the previous literature. The relationship between these criteria and the results obtained in the present paper is clarified. Some simple applications of the Sugeno's integral semantics of quantifiers are presented.

Keywords: High level knowledge representation and reasoning; Natural language understanding; Computing with words; Fuzzy logic; Quantifier; Fuzzy measure; Sugeno's integral

1. Introduction

First order logic increases the expressive power of propositional logic a lot through adding quantifiers. Classical first order logic only possesses two quantifiers, the universal quantifier ($\forall$) and the existential quantifier ($\exists$). However, these quantifiers are often too limited to express some properties of certain mathematical structures and to model certain knowledge stated in natural languages. This leads logicians and linguists to introduce the notion of generalized quantifiers.

As early as in 1957, Mostowski [29] proposed a general notion of generalized quantifier and showed that first order logic with a class of generalized quantifiers are not axiomatizable. This work together with others initiated the subject of model theoretic logics.

Barwise and Cooper [2] started the studies of generalized quantifiers in natural languages. Since then, a rich variety of generalized quantifiers in natural languages have been found, and their expressive power and logical properties have been thoroughly investigated from both semantic and syntactic aspects. In particular, van Benthem [36] viewed

E-mail address: yingmsh@tsinghua.edu.cn (M. Ying).

[1] This work was partly supported by the National Foundation of Natural Sciences of China (Grant No: 60321002, 60496321) and the Key Grant Project of Chinese Ministry of Education (Grant No: 10403).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2006.02.001

a generalized quantifier as a relation on the subsets of a universe of discourse, and systematically examined various relational behaviors of generalized quantifiers such as reflexivity, symmetry, transitivity, linearity and monotonicity and their roles in realizing certain inference patterns. For a recent review on the theory of generalized quantifiers in natural languages, we refer to [24].

It has been clearly realized in the artificial intelligence community that natural languages are suited to high level knowledge representation [26,33]. This is indeed one of the main motivations of computing with words [25,50,57]. However, classical logic is not adequate to face the essential uncertainty, vagueness and ambiguity of human reasoning expressed in natural languages. Consequently, the logical treatments and mathematical models of the concepts of uncertainty, vagueness and ambiguity is of increasing importance in artificial intelligence and related researches, and many logicians have proposed different logic systems as a formalization of reasoning under uncertainty, vagueness and ambiguity (see, for example, [3,44–47,49], [11, Chapter III.1], or [19, Chapter 7]).

Since quantifiers have the ability of summarizing the properties of a class of objects without enumerating them, linguistic quantification is a very important topic in the field of knowledge representation and reasoning. Quantifiers in natural languages are usually vague in some sense. Some representative examples of linguistic quantifiers with vagueness are [56]: *several, most, much, not many, very many, not very many, few, quite a few, large number, small number, close to five, approximately ten, frequently*. It is clear that two-valued logic is not suited to cope with vague quantifiers. There has been, therefore, increasing interest about logical treatment of quantifiers in human languages in fuzzy logic community. Indeed, sometimes fuzzy logic permits a more precise representation of the kind of quantifiers in various natural languages.

The first fuzzy set theoretic approach to linguistic quantifiers was described by Zadeh [55,56]. In his approach, linguistic quantifiers are treated as fuzzy numbers and they may be manipulated through the use of arithmetic for fuzzy numbers. The truth evaluation of a linguistically quantified statement is performed by computing the cardinality of the fuzzy set defined by the linguistic predicate in such a statement and then by finding the degree to which this cardinality is compatible with the involved quantifier. Since then, a considerable amount of literature [1,4–7,9,10, 12,15–18,30–32,42,43,51,52] has been devoted to the studies of linguistic quantifiers in the framework of fuzzy set theory. For example, in a series of papers [39–41], Yager proposed the substitution method for evaluating quantified propositions and the method based on OWA operators. For a survey, see [27,28].

On the other hand, fuzzy quantification models are employed in solving a great variety of problems from many different fields such as database querying [7,23], data mining and knowledge discovering [7,25], information fusion [21,25], group decision making and multiple-objective decision making [20,40], inductive learning [21], and optimization and control [22].

This paper introduces a new framework for modeling quantifiers in natural languages. In this framework, linguistic quantifiers are represented by Sugeno's fuzzy measures [35]. More precisely, a quantifier Q is seen as a family of fuzzy measures indexed by nonempty sets. For each nonempty set X, the quantifier Q limited to the discourse universe X is defined to be a fuzzy measure Q_X on X, and for any subset E of X, the quantity $Q_X(E)$ expresses the truth value of the quantified statement "Q *Xs are As*" when A is a crisp predicate and the set of elements in X satisfying A is E. As is well known, predicates in linguistically quantified statements are often vague too. In this general case, the truth value of a quantified proposition is then evaluated by using Sugeno's integral [35].

The advantage of this framework is that it allows us to have some elegant logical properties of linguistic quantifiers. For example, we are able to establish a prenex normal form theorem for linguistic quantifiers (see Corollary 34). It should be pointed out that this paper only deals with increasing quantifiers because fuzzy measures assume monotonicity. Thus, quantifiers such as *several, few, quite a few, small number, not many, not very many, close to five, approximately ten* cannot be modeled in our proposed setting.

This paper is arranged as follows. For convenience of the reader, in Section 2 we review some notions and results from the theory of Sugeno's fuzzy measures and integrals. In Section 3, (linguistic) quantifiers are formally defined in terms of fuzzy measure, and several operations of quantifiers are introduced. In Section 4, we construct a first order language with linguistic quantifiers and present semantics of such a logical language. In particular, the truth valuation of quantified formulas is given by using Sugeno's integrals. Section 5 is devoted to examine thoroughly logical properties of linguistic quantifiers. In particular, we prove a prenex normal form theorem for logical formulas with linguistic quantifiers. In Section 6, the notions of cardinal and numeric quantifiers are introduced so that we are able to establish a close link between the Sugeno integral semantics and the Zadeh's cardinality-based semantics of linguistic quantifiers. In Section 7, we present some simple applications to illustrate the utility of the results obtained

in the current paper. In Section 8, our Sugeno integral approach to evaluation of quantified statements is compared with others. A set of criteria for linguistic quantification was proposed in the previous literature. The relationship between these criteria and the results obtained in the present paper is clarified. We draw conclusions and point out some problems for further studies in Section 9.

2. Fuzzy measures and Sugeno integrals

This is a preliminary section. In this section, we are going to review some notions and fundamental results needed in the sequel from the theory of fuzzy measures and Sugeno's integrals. For details, we refer to [35] or [11, Chapter 5].

The theory of fuzzy measures and integrals was originally proposed by Sugeno [35]. Fuzzy measure is a generalization of the notion of measure in mathematical analysis, and it relaxes the condition of additivity for usual measure and only assume monotonicity. Thus, fuzzy measures are very general, and probability measures, Zadeh's possibility measures, Shafer's belief functions among others [37] are shown to be special cases of fuzzy measures. Sugeno's integral is analogous to Lebesgue integral. The difference between them is that addition and multiplication in the definition of Lebesgue integral are replaced respectively by the operations "min" and "max" when Sugeno's integral is considered. Since its inception, the theory of Sugeno's measures and integrals has been applied in the fields of subjective evaluation, decision systems and pattern recognition, name a few.

A fuzzy measure on a set X is a function defined on some subsets of X. In general, the domain of a fuzzy measure can be taken to be a monotone family which is a set of subsets of X containing $\emptyset$ and X itself and closed under limits of monotone sequences of subsets of X. But in this paper we focus our attention on a special class of monotone family called Borel field.

Definition 1. [35, page 10] Let X be a nonempty set. A *Borel field* over X is a subset $\wp$ of 2^X satisfying the next conditions:

(1) $\emptyset \in \wp$;
(2) If $E \in \wp$, then $X - E \in \wp$; and
(3) If $E_n \in \wp$ for $1 \leqslant n < \infty$, then $\bigcup_{n=1}^{\infty} E_n \in \wp$.

A typical example of Borel field over a nonempty set X is the power set 2^X of X. Indeed, in this paper, we will mainly consider this special Borel field.

Definition 2. [35, Definition 2.3] If X is a nonempty set and $\wp$ is a Borel field over X, then $(X, \wp)$ is called a *measurable space*.

In order to define fuzzy measure, we need the notion of limit of set sequence. If $E_1 \subseteq \cdots \subseteq E_n \subseteq E_{n+1} \subseteq \cdots$, then the sequence $\{E_n\}$ is said to be increasing and we define

$$\lim_{n\to\infty} E_n = \bigcup_{n=1}^{\infty} E_n,$$

and if $E_1 \supseteq \cdots \supseteq E_n \supseteq E_{n+1} \supseteq \cdots$, then the sequence $\{E_n\}$ is said to be decreasing and we define

$$\lim_{n\to\infty} E_n = \bigcap_{n=1}^{\infty} E_n.$$

Both increasing and decreasing sequences of sets are said to be monotone.

Definition 3. [35, Definitions 2.2 and 2.4] Let $(X, \wp)$ be a measurable space. If a set function $m : \wp \to [0, 1]$ satisfies the following properties:

(1) $m(\emptyset) = 0$ and $m(X) = 1$;
(2) (Monotonicity) If $E, F \in \wp$ and $E \subseteq F$, then $m(E) \leqslant m(F)$; and

(3) (Continuity) If $E_n \in \wp$ for $1 \leqslant n < \infty$ and $\{E_n\}$ is monotone, then

$$m\Big(\lim_{n\to\infty} E_n\Big) = \lim_{n\to\infty} m(E_n),$$

then m is called a *fuzzy measure* over $(X, \wp)$, and $(X, \wp, m)$ is called a *fuzzy measure space*.

Intuitively, $m(E)$ expresses someone's subjective evaluation of the statement "x is in E" in a situation in which he guesses whether x is in E.

The continuity of fuzzy measure is often discarded. Obviously, a probability measure is a fuzzy measure. The notion of plausibility measure introduced in [13,14] or [19] (Section 2.8) is similar to that of fuzzy measure. The only difference between them is that the range of a plausibility measure can be any partially ordered set with top and bottom elements rather than the unit interval.

Let us first consider an example of fuzzy measure which will be used to define the existential quantifier.

Example 4. Let X be a nonempty set. If $\pi : X \to [0, 1]$ is a mapping with $\sup_{x\in X} \pi(x) = 1$, then it is called a possibility distribution, and for each $E \subseteq X$, we define

$$\Pi_\pi(E) = \sup_{x\in E} \pi(x).$$

Then $\Pi_\pi(\cdot)$ is a fuzzy measure over $(X, 2^X)$ and it is called the possibility measure induced by π. It should be noted that a possibility measure is not continuous from top, that is, there is a decreasing sequence $\{E_n\}$ with

$$\Pi_\pi\Big(\lim_{n\to\infty} E_n\Big) < \lim_{n\to\infty} \Pi_\pi(E_n).$$

(1) Suppose that $x_0 \in X$ and

$$\pi_{x_0}(x) = \begin{cases} 1, & \text{if } x = x_0, \\ 0, & \text{otherwise.} \end{cases}$$

Then π_{x_0} is a possibility distribution and $\Pi_{\pi_{x_0}}$ is the membership function of x_0, that is, for any $E \subseteq X$,

$$\Pi_{\pi_{x_0}}(E) = \begin{cases} 1, & \text{if } x_0 \in E, \\ 0, & \text{otherwise.} \end{cases}$$

(2) If we define $\pi(x) = 1$ for all $x \in X$, then it is easy to see that for any $E \subseteq X$,

$$\Pi_\pi = \begin{cases} 1, & \text{if } E \neq \emptyset, \\ 0, & \text{otherwise.} \end{cases}$$

The notion of dual fuzzy measure is required when dealing with dual linguistic quantifier. We introduce such a notion in the next definition.

Definition 5. Let $(X, \wp, m)$ be a fuzzy measure space. Then the dual set function $m^* : \wp \to [0, 1]$ of m is defined by

$$m^*(E) = 1 - m(X - E)$$

for each $E \in \wp$.

It is easy to see that m^* is a fuzzy measure over $(X, \wp)$ too.

The following example gives the dual of possibility measure, necessity measure. It will be used in defining the universal quantifier.

Example 6. Let π be a possibility distribution over X and Π_π the possibility measure induced by π. Then the dual measure $N_\pi = \Pi_\pi^*$ of Π_π is given by

$$N_\pi(E) = \inf_{x\notin E} \big(1 - \pi(x)\big)$$

for each $E \subseteq X$, and it is called the necessity measure induced by π.

(1) It is interesting to note that $N_{\pi_{x_0}} = \Pi^*_{\pi_{x_0}} = \Pi_{\pi_{x_0}}$.
(2) If $\pi(x) = 1$ for every $x \in X$, then

$$N_\pi(E) = \begin{cases} 1, & \text{if } E = X, \\ 0, & \text{otherwise.} \end{cases}$$

We now turn to consider the Sugeno's integral of a function. It is not the case that every function is integrable. Those integrable functions are isolated by the following definition.

Definition 7. [35, Definition 3.3] Let $(X, \wp)$ be a measurable space, and let $h : X \to [0, 1]$ be a mapping from X into the unit interval. For any $\lambda \in [0, 1]$, we write

$$H_\lambda = \big\{x \in X\colon\ h(x) \geqslant \lambda\big\}.$$

If $H_\lambda \in \wp$ for all $\lambda \in [0, 1]$, then h is said to be *$\wp$-measurable*.

The following lemma demonstrates measurability of some composed functions. Indeed, we will see later that this measurability guarantees that truth values of quantifications of negation, conjunction and disjunction are well-defined in our Sugeno integral semantics.

Lemma 8. [35, Propositions 3.1 and 3.2] *If h, h_1 and $h_2 : X \to [0, 1]$ are all $\wp$-measurable functions, then so are $1 - h$, $\min(h_1, h_2)$ and $\max(h_1, h_2)$, where*

$$(1 - h)(x) = 1 - h(x),$$
$$\min(h_1, h_2)(x) = \min\big(h_1(x), h_2(x)\big)$$

and

$$\max(h_1, h_2)(x) = \max\big(h_1(x), h_2(x)\big)$$

for every $x \in X$.

Now we are able to present the key notion in this section.

Definition 9. [35, Definition 3.1 and page 19] Let $(X, \wp, m)$ be a fuzzy measure space. If $A \in \wp$ and $h : X \to [0, 1]$ is a $\wp$-measurable function, then the *Sugeno's integral* of h over A is defined by

$$\int_A h \circ m = \sup_{\lambda \in [0,1]} \min\big[\lambda, m(A \cap H_\lambda)\big],$$

where $H_\lambda = \{x \in X\colon\ h(x) \geqslant \lambda\}$ for each $\lambda \in [0, 1]$. In particular, $\int_A h \circ m$ will be abbreviated to $\int h \circ m$ whenever $A = X$.

The next lemma gives an alternative definition of Sugeno's integral for the case that the Borel field in a measurable space is taken to be the power set of the underlying set.

Lemma 10. [35, Theorem 3.1] *If the Borel field $\wp$ in the fuzzy measure space $(X, \wp, m)$ is the power set 2^X of X, then for any function $h : X \to [0, 1]$, we have*:

$$\int_A h \circ m = \sup_{F \in 2^X} \min\big[\inf_{x \in F} h(x), m(A \cap F)\big].$$

A simplified calculation method of Sugeno's integrals over finite sets is presented in the following lemma.

Lemma 11. [35, Theorem 4.1] *Let $X = \{x_1, \ldots, x_n\}$ be a finite set, and let $h :\to [0, 1]$ be such that $h(x_i) \leqslant h(x_{i+1})$ for $1 \leqslant i \leqslant n-1$ (if not so, rearrange $h(x_i)$, $1 \leqslant i \leqslant n$). Then*

$$\int_A h \circ m = \max_{i=1}^{n} \min\big[h(x_i), m(A \cap X_i)\big],$$

where $X_i = \{x_j : i \leqslant i \leqslant n\}$ for $1 \leqslant i \leqslant n$.

We now collect some properties of Sugeno's integrals needed in what follows.

Lemma 12. [35, Propositions 3.4 and 3.5, Theorems 3.2 and 3.5] *Suppose that $(X, \wp, m)$ is a fuzzy measure space.*

(1) *Let $a \in [0, 1]$. Then it holds that*

$$\int a \circ m = a,$$

where "a" in the left-hand side is seen as a constant function $a : X \to [0, 1]$ such that $a(x) = a$ for every $x \in X$.

(2) *Let $h, h' : X \to [0, 1]$ be two $\wp$-measurable functions. If $h \leqslant h'$, then there holds*

$$\int h \circ m \leqslant \int h' \circ m.$$

Moreover, for any $\wp$-measurable functions $h_1, h_2 :\to [0, 1]$, we have:

$$\int \max(h_1, h_2) \circ m \geqslant \max\left(\int h_1 \circ m, \int h_2 \circ m\right),$$

$$\int \min(h_1, h_2) \circ m \leqslant \min\left(\int h_1 \circ m, \int h_2 \circ m\right).$$

In particular, the first inequality becomes equality when m is a possibility measure.

(3) *Let $a \in [0, 1]$ and let $h : X \to [0, 1]$ be a $\wp$-measurable function. Then there hold*

$$\int \max(a, h) \circ m = \max\left(a, \int h \circ m\right),$$

$$\int \min(a, h) \circ m = \min\left(a, \int h \circ m\right),$$

where "a" in the left-hand side is as in (1).

It may be observed that fuzzy measures are defined over crisp sets before. To conclude this section, we introduce the notion of extension of fuzzy measure over fuzzy sets. Zadeh [54] introduced a natural extension of probability measure on fuzzy sets. In a similar manner, an extension of fuzzy measure on fuzzy sets can be defined.

Definition 13. [35, Definition 3.7] Let $(X, \wp, m)$ be a fuzzy measure space and let $\widetilde{\wp}$ be the set of fuzzy subsets of X with $\wp$-measurable membership functions. Then the extension $\widetilde{m}$ of m on $\widetilde{\wp}$ is defined by

$$\widetilde{m}(h) = \int h \circ m$$

for all $h \in \widetilde{\wp}$.

3. Fuzzy quantifiers

Many versions of fuzzy set theoretic definition of linguistic quantifier have been introduced in the previous literature. In this paper, we take a different starting point, and a linguistic quantifier will be represented by a family of fuzzy measures. We first give a formal definition of fuzzy quantifier in this new framework. To do this, a new notation is needed. For any measurable space $(X, \wp)$, we write $M(X, \wp)$ for the set of all fuzzy measures on $(X, \wp)$.

Definition 14. A fuzzy quantifier (or quantifier for short) consists of the following two items:

(i) for each nonempty set X, a Borel field $\wp_X$ over X is equipped; and

(ii) a choice function

$$Q : (X, \wp_X) \mapsto Q_{(X,\wp_X)} \in M(X, \wp_X)$$

of the (proper) class $\{M(X, \wp_X) : (X, \wp_X)$ is a measurable space$\}$.

Intuitively, for a given discourse universe X, if the set of objects in X satisfying a (crisp) property A is E, then the quantity $Q_X(E)$ is thought of as the truth value of the quantified proposition "*Q Xs are As*".

For simplicity, $Q_{(X,\wp_X)}$ is often abbreviated to Q_X whenever the Borel field $\wp_X$ can be recognized from the context. In some applications, Q_X is allowed to be undefined or unspecified for some sets X.

To illustrate the above definition, let us consider some examples. The simplest quantifiers are the universal and existential quantifiers.

Example 15. The universal quantifier $\forall$ = "all" and the existential quantifier $\exists$ = "some" are defined as follows, respectively: for any set X and for any $E \subseteq X$,

$$\forall_X(E) = \begin{cases} 1, & \text{if } E = X, \\ 0, & \text{otherwise;} \end{cases}$$

$$\exists_X(E) = \begin{cases} 1, & \text{if } E \neq \emptyset, \\ 0, & \text{otherwise.} \end{cases}$$

The universal and existential quantifiers are crisp quantifiers because $\forall_X(E), \exists_X(E) \in \{0, 1\}$ for all $E \subseteq X$.

This example shows that both the universal and existential quantifiers can be accommodated in our fuzzy measure definition of quantifiers. However, when the discourse universe X is infinite, Zadeh's cardinality approach to quantifiers cannot be used to treat the universal quantifier because it is possible that a proper subset E of X has the same cardinality as X.

Except the universal and existential quantifiers, some quantifiers frequently occurred in natural languages can also be defined well in terms of fuzzy measures. Let us see the following example.

Example 16. For any set X and for any $E \subseteq X$, we define

$$\textit{at least three}_X(E) = \begin{cases} 1, & \text{if } |E| \geqslant 3, \\ 0, & \text{otherwise;} \end{cases}$$

"at least three" is an example of crisp generalized quantifier too. The following are three typical examples of fuzzy quantifiers. Suppose that X is a nonempty finite set. Then we define

$$\textit{many}_X(E) = \frac{|E|}{|X|},$$

$$\textit{most}_X(E) = \left(\frac{|E|}{|X|}\right)^{3/2},$$

$$\textit{almost all}_X(E) = \left(\frac{|E|}{|X|}\right)^{2}$$

for any subset E of X, where $|E|$ stands for the cardinality of E. The above definitions of quantifiers "many", "most" and "almost all" can be generalized to the case of infinite discourse universe X. Let $(X, \wp)$ be a measurable space, and let μ be a finite measure on $(X, \wp)$, that is, a mapping $\mu : \wp \to [0, \infty)$ such that

$$\mu\left(\bigcup_{n=1}^{\infty} E_n\right) = \sum_{n=1}^{\infty} \mu(E_n),$$

whenever $\{E_n\}_{n=1}^{\infty}$ is a countable pairwise disjoint sub-family of $\wp$. Then we may define

$$many_X(E) = \frac{\mu(E)}{\mu(X)},$$
$$most_X(E) = \left(\frac{\mu(E)}{\mu(X)}\right)^{3/2},$$
$$almost\ all_X(E) = \left(\frac{\mu(E)}{\mu(X)}\right)^{2}$$

for any $E \in \wp$.

The above definitions for quantifiers "many", "most" and "almost all" seem to be largely arbitrary except the ordering: $almost\ all_X(E) \leqslant most_X(E) \leqslant many_X(E)$ for any $E \subseteq X$. One may naturally ask the question: how we can avoid this arbitrariness, or more generally, is there any general methodology for defining linguistic quantifiers? This is in a sense equivalent to the problem of how to derive the membership function of a fuzzy set, which is arguably the biggest open problem in fuzzy set theory. In the 1970s quite a few approaches to estimation of membership functions had been proposed, including psychological analysis, statistics and preference method. For a survey on these early works, see [11, Chapter IV.1]. However, no significant further progress has been made in more than 30 years. The current situation is that a theoretical foundation for membership function estimation is still missing. In particular, we do not have a mathematical theorem in fuzzy set theory which guarantees that the membership function of a fuzzy set can be approximated in some way, like the law of large numbers in probability theory.

Two kinds of fuzzy quantifiers are of special interest, and they are defined in the next definition. We will see later that they enjoy some very nice logical properties.

Definition 17. A quantifier Q is called a possibility quantifier (resp. necessity quantifier) if for any nonempty set X, and for any $E_1, E_2 \in \wp_X$,

$$Q_X(E_1 \cup E_2) = \max\big(Q_X(E_1), Q_X(E_2)\big)$$
$$\big(\text{resp. } Q_X(E_1 \cap E_2) = \min\big(Q_X(E_1), Q_X(E_2)\big)\big).$$

It is clear that the universal quantifier ($\forall$) and the existential quantifier ($\exists$) are respectively a necessity quantifier and a possibility quantifier. More generally, if for each set X, Q_X is a possibility (resp. necessity) measure induced by some possibility distribution on X, then Q is a possibility (resp. necessity) quantifier.

The following definition introduces a partial order between fuzzy quantifiers and three operations of fuzzy quantifiers.

Definition 18. Let Q, Q_1 and Q_2 be quantifiers. Then

(1) We say that Q_1 is stronger than Q_2, written $Q_1 \sqsubseteq Q_2$, if for any nonempty set X and for any $E \in \wp_X$, we have $Q_{1X}(E) \leqslant Q_{2X}(E)$.

(2) The dual Q^* of Q, and the meet $Q_1 \sqcap Q_2$ and union $Q_1 \sqcup Q_2$ of Q_1 and Q_2 are defined respectively as follows: for any nonempty set X and for any $E \in \wp_X$,

$$Q^*_X(E) \stackrel{\text{def}}{=} 1 - Q_X(X - E),$$
$$(Q_1 \sqcap Q_2)_X(E) \stackrel{\text{def}}{=} \min\big(Q_{1X}(E), Q_{2X}(E)\big),$$
$$(Q_1 \sqcup Q_2)_X(E) \stackrel{\text{def}}{=} \max\big(Q_{1X}(E), Q_{2X}(E)\big).$$

It may be observed that the meet and union operations of quantifiers are exactly the set-theoretic intersection and union operations, respectively, when quantifiers are imagined as fuzzy subsets of the given Borel field $\wp_X$ over an universe X of discourse. The intersection and union of fuzzy sets were first defined by Zadeh [53] in terms of "min" and "max" operations on membership functions. Afterwards, many different intersection and union operations of fuzzy sets have been proposed in the literature (see for example [11, Section II.1.B]). Indeed, all of these operations can be unified in the framework of t-norm and t-conorm, two notions initially introduced in the theory of probabilistic metric spaces [34]. This observation suggests the possibility of replacing the "min" and "max" operations in the defining

equations of $Q_1 \sqcap Q_2$ and $Q_1 \sqcup Q_2$ by a general t-norm and t-conorm, respectively. When once such more general operations of quantifiers are adopted, the logical properties of linguistic quantifier related to the meet and union operations of quantifiers obtained in this paper should naturally be reexamined (cf. the remark after Proposition 31). On the other hand, in practical applications an important problem is how to choose suitable t-norms and t-conorms. There have been some experiments verifying the accurateness of various fuzzy set operations reported in the literature. For a short survey on the early works in this direction, see [11, Section IV.1.C].

Example 19. (1) The universal and existential quantifiers are the dual of each other: $\forall^* = \exists$ and $\exists^* = \forall$.

(2) For any nonempty finite set X and for any $E \subseteq X$,

$$(\textit{almost all})^*_X(E) = \frac{2|E|}{|X|} - \left(\frac{|E|}{|X|}\right)^2,$$

$$(\textit{at least three} \sqcap \textit{many})_X(E) = \begin{cases} \frac{|E|}{|X|}, & \text{if } |E| \geqslant 3, \\ 0, & \text{otherwise,} \end{cases}$$

$$(\textit{at least three} \sqcup \textit{many})_X(E) = \begin{cases} 1, & \text{if } |E| \geqslant 3, \\ \frac{2}{|X|}, & \text{if } |E| = 2, \\ \frac{1}{|X|}, & \text{if } |E| = 1, \\ 0, & \text{if } E = \emptyset. \end{cases}$$

Several algebraic laws for quantifier operations are presented in the next lemma. They show that quantifiers together with the operations defined above form a De Morgan algebra.

Lemma 20. (1) *For any quantifier Q, we have $\forall \sqsubseteq Q \sqsubseteq \exists$. In other words, the universal quantifier* ($\forall$) *is the strongest quantifier, and the existential quantifier* ($\exists$) *is the weakest one.*

(2) *For all quantifiers Q_1 and Q_2, it holds that $Q_1 \sqcap Q_2 \sqsubseteq Q_1$ and $Q_1 \sqsubseteq Q_1 \sqcup Q_2$.*

(3) *For all quantifiers Q_1 and Q_2, we have*:

(*Commutativity*) $Q_1 \sqcap Q_2 = Q_2 \sqcap Q_1$, $Q_1 \sqcup Q_2 = Q_2 \sqcup Q_1$.

(*Associativity*) $Q_1 \sqcap (Q_2 \sqcap Q_3) = (Q_1 \sqcap Q_2) \sqcap Q_3$, $Q_1 \sqcup (Q_2 \sqcup Q_3) = (Q_1 \sqcup Q_2) \sqcup Q_3$.

(*Absorption*) $Q_1 \sqcap (Q_1 \sqcup Q_2) = Q_1$, $Q_1 \sqcup (Q_1 \sqcap Q_2) = Q_1$.

(*De Morgan law*) $(Q_1 \sqcap Q_2)^* = Q_1^* \sqcup Q_2^*$, $(Q_1 \sqcup Q_2)^* = Q_1^* \sqcap Q_2^*$.

Proof. Immediate from Definition 18. □

4. A first order language with linguistic quantifiers and its semantics

We first construct a first order logical language $\mathbf{L_q}$ with linguistic quantifiers. The alphabet of our language $\mathbf{L_q}$ is given as follows:

(1) A denumerable set of individual variables: $x_0, x_1, x_2, \ldots$;

(2) A set $\mathbf{F} = \bigcup_{n=0}^{\infty} \mathbf{F}_n$ of predicate symbols, where $\mathbf{F}_n$ is the set of all n-place predicate symbols for each $n \geqslant 0$. It is assumed that $\bigcup_{n=1}^{\infty} \mathbf{F}_n \neq \emptyset$;

(3) Propositional connectives: $\sim$, $\wedge$; and

(4) Parentheses: (,).

The syntax of the language $\mathbf{L_q}$ is then presented by the following definition.

Definition 21. The set Wff of well-formed formulas is the smallest set of symbol strings satisfying the following conditions:

(i) If $n \geqslant 0$, $F \in \mathbf{F}_n$, and $y_1, \ldots, y_n$ are individual variables, then $F(y_1, \ldots, y_n) \in$ Wff;

(ii) If Q is a quantifier, x is an individual variable, and $\varphi \in$ Wff, then $(Qx)\varphi \in$ Wff; and

(iii) If $\varphi, \varphi_1, \varphi_2 \in$ Wff, then $\sim \varphi$, $\varphi_1 \wedge \varphi_2 \in$ Wff.

For the sake of simplicity, we introduce some abbreviations:

$$\varphi \vee \psi \stackrel{\text{def}}{=} \sim (\sim \varphi \wedge \sim \psi),$$
$$\varphi \to \psi \stackrel{\text{def}}{=} \sim \varphi \vee \psi,$$
$$\varphi \leftrightarrow \psi \stackrel{\text{def}}{=} (\varphi \to \psi) \wedge (\psi \to \varphi).$$

The notions of bound variable and free variable can be introduced in a standard way. We omit their detailed definitions here but freely use them in the sequel.

The semantics of our language $\mathbf{L_q}$ is given by the next two definitions.

Definition 22. An interpretation I of our logical language consists of the following items:

(i) A measurable space $(X, \wp)$, called the domain of I;

(ii) For each $n \geqslant 0$, we associate the individual variable x_i with an element x_i^I in X; and

(iii) For any $n \geqslant 0$ and for any $F \in \mathbf{F}_n$, there is a $\wp^n$-measurable function $F^I : X^n \to [0, 1]$.

For simplicity, in what follows we assume that the Borel field $\wp$ in the domain $(X, \wp)$ of an interpretation I is always taken to be the power set 2^X of X, and the Borel field $\wp_X$ equipped with a nonempty set X is also 2^X for any quantifier Q. This often shortens the presentation and proof of our results.

Definition 23. Let I be an interpretation. Then the truth value $T_I(\varphi)$ of a formula φ under I is defined recursively as follows:

(i) If $\varphi = F(y_1, \ldots, y_n)$, then

$$T_I(\varphi) = F^I\big(y_1^I, \ldots, y_n^I\big).$$

(ii) If $\varphi = (Qx)\psi$, then

$$T_I(\varphi) = \int T_{I\{./x\}}(\psi) \circ Q_X,$$

where X is the domain of I, $T_{I\{./x\}}(\psi) : X \to [0, 1]$ is a mapping such that

$$T_{I\{./x\}}(\varphi)(u) = T_{I\{u/x\}}(\varphi)$$

for all $u \in X$, and $I\{u/x\}$ is the interpretation which differs from I only in the assignment of the individual variable x, that is, $y^{I\{u/x\}} = y^I$ for all $y \neq x$ and $x^{I\{u/x\}} = u$;

(iii) If $\varphi = \sim \psi$, then

$$T_I(\varphi) = 1 - T_I(\psi),$$

and if $\varphi = \varphi_1 \wedge \varphi_2$, then

$$T_I(\varphi) = \min\big(T_I(\varphi_1), T_I(\varphi_2)\big).$$

The following proposition establishes a close link between the truth evaluation of quantified statement and the extension of fuzzy measure on fuzzy sets.

Proposition 24. *Let Q be a quantifier and x an individual variable, and let $\varphi \in$ Wff. Then for any interpretation I,*

$$T_I\big((Qx)\varphi\big) = \widetilde{Q_X}\big(T_I(\varphi)\big),$$

where $\widetilde{Q_X}$ is the extension of Q_X on fuzzy sets.

Proof. Immediate from Definitions 13 and 23(ii). □

In order to illustrate further the evaluation mechanism of quantified propositions, let us examine some simple examples.

Example 25. We first consider the simplest case of quantification. For any quantifier Q and for any $\varphi \in \mathrm{Wff}$, if I is an interpretation with the domain being a singleton $X = \{u\}$, then for any quantifier Q,

$$T_I\big((Qx)\varphi\big) = T_I(\varphi).$$

This means that quantification degenerates on a singleton discourse universe.

Example 26. We now consider the strongest and weakest quantifiers. This example shows that the Sugeno integral evaluation of universally and existentially quantified statements coincide with the standard way, and so gives a witness for reasonableness of Sugeno integral semantics of linguistic quantification. Let Q be a quantifier and x an individual variable, and let $\varphi \in \mathrm{Wff}$. Then for any interpretation I with domain X, we have

$$\begin{aligned} T_I\big((\exists x)\varphi\big) &= \int T_{I\{u/x\}}(\varphi) \circ \exists_X = \sup_{F\subseteq X} \min\big[\inf_{u\in F} T_{I\{u/x\}}(\varphi), \exists_X(F)\big] \\ &= \sup_{\emptyset\neq F\subseteq X} \inf_{u\in F} T_{I\{u/x\}}(\varphi) = \sup_{u\in X} T_{I\{u/x\}}(\varphi). \end{aligned}$$

Similarly, it holds that

$$T_I\big((\forall x)\varphi\big) = \inf_{u\in X} T_{I\{u/x\}}(\varphi).$$

To conclude this section, in the case of finite discourse universe we give a necessary and sufficient condition under which the truth value of a quantified proposition is bound by a given threshold value from up or below. This condition is very useful in some real applications (see Example 43 below).

Proposition 27. *Let X be a finite set, let I be an interpretation with X as its domain, and let $\lambda \in [0,1]$. Then for any quantifier Q and $\varphi \in \mathrm{Wff}$, we have*:

(i) $T_I((Qx)\varphi) \geqslant \lambda$ *if and only if*

$$Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \lambda\}\big) \geqslant \lambda.$$

(ii) $T_I((Qx)\varphi) \leqslant \lambda$ *if and only if*

$$Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi) > \lambda\}\big) \leqslant \lambda.$$

Proof. (i) If $Q_X(\{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \lambda\}) \geqslant \lambda$, then we obtain

$$\begin{aligned} T_I\big((Qx)\varphi\big) &= \sup_{\mu\in[0,1]} \min\big(\mu, Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \mu\}\big)\big) \\ &\geqslant \min\big(\lambda, Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \lambda\}\big)\big) \geqslant \lambda. \end{aligned}$$

Conversely, if

$$T_I\big((Qx)\varphi\big) = \sup_{F\subseteq X} \min\big(\inf_{u\in F} T_{I\{u/x\}}(\varphi), Q_X(F)\big) \geqslant \lambda,$$

then there exists $F_0 \subseteq X$ such that

$$\min\big(\inf_{u\in F_0} T_{I\{u/x\}}(\varphi), Q_X(F_0)\big) \geqslant \lambda$$

because X is finite. Then it holds that $\inf_{u\in F_0} T_{I\{u/x\}}(\varphi) \geqslant \lambda$ and $Q_X(F_0) \geqslant \lambda$. Furthermore, we have $T_{I\{u/x\}}(\varphi) \geqslant \lambda$ for every $u \in F_0$ and $\{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \lambda\} \supseteq F_0$. This yields

$$Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \lambda\}\big) \geqslant Q_X(F_0) \geqslant \lambda.$$

(ii) We first prove the "if" part. If $\mu \leqslant \lambda$, then it is obvious that

$$\min\big[\mu, Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \mu\}\big)\big] \leqslant \lambda,$$

and if $\mu > \lambda$, then $\{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \mu\} \subseteq \{u \in X\colon T_{I\{u/x\}}(\varphi) > \lambda\}$,

$$Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \mu\}\big) \leqslant Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi) > \lambda\}\big) \leqslant \lambda,$$

and we also have

$$\min\big[\mu, Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \mu\}\big)\big] \leqslant \lambda.$$

Thus,

$$T_I\big((Qx)\varphi\big) = \sup_{\mu\in[0,1]} \min\big[\mu, Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \mu\}\big)\big] \leqslant \lambda.$$

For the "only if" part, suppose that

$$\mu_0 = \inf\big\{T_{I\{u/x\}}(\varphi)\colon T_{I\{u/x\}}(\varphi) > \lambda\big\}.$$

Since X is finite, it holds that $\mu_0 > \lambda$ and $\{u \in X\colon T_{I\{u/x\}}(\varphi) > \lambda\} = \{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \mu_0\}$. Consequently,

$$\begin{aligned}\min\big[\mu_0, Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi) > \lambda\}\big)\big] &= \min\big[\mu_0, Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \mu_0\}\big)\big] \\ &\leqslant \sup_{\mu\in[0,1]} \min\big[\mu, Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi) \geqslant \mu\}\big)\big] \\ &= T_I\big((Qx)\varphi\big) \leqslant \lambda.\end{aligned}$$

Finally, from $\mu_0 > \lambda$ we know that $Q_X(\{u \in X\colon T_{I\{u/x\}}(\varphi) > \lambda\}) \leqslant \lambda$. □

5. Logical properties of linguistic quantifiers

The main purpose of this section is to establish various logical properties of linguistic quantifiers. In order to present these properties in a compact way, several meta-logical notions are needed.

Definition 28. Let $\varphi \in \mathrm{Wff}$ and $\Sigma \subseteq \mathrm{Wff}$.

(1) If for any interpretation I, $T_I(\varphi) \geqslant \frac{1}{2}$, then φ is said to be fuzzily valid and we write $\models^{Fuz} \varphi$.

(2) If for any interpretation I,

$$\inf_{\psi\in\Sigma} T_I(\psi) \leqslant T_I(\varphi),$$

then φ is called a consequence of Σ and we write $\Sigma \models \varphi$. In particular, if $\emptyset \models \varphi$, that is, $T_I(\varphi) = 1$ for each interpretation I, then φ is said to be (absolutely) valid and we write $\models \varphi$.

(3) If $\varphi \models \psi$ and $\psi \models \varphi$, that is, for any interpretation I, $T_I(\varphi) = T_I(\psi)$, then we say that φ and ψ are equivalent and write $\varphi \equiv \psi$.

We first consider the question how the consequence relation is preserved by quantifiers and the question when can quantifiers distribute over conjunctions and disjunctions. The following proposition answers these two questions.

Proposition 29. (1) *Suppose that* $\varphi_1, \varphi_2 \in \mathrm{Wff}$. *Then* $\varphi_1 \models \varphi_2$ *if and only if for any quantifier* Q *and for any individual variable* x, $(Qx)\varphi_1 \models (Qx)\varphi_2$ *always holds.*

(2) *For any quantifier* Q *and for any* $\varphi_1, \varphi_2 \in \mathrm{Wff}$, *we have*:

$$(Qx)(\varphi_1 \wedge \varphi_2) \models (Qx)\varphi_1 \wedge (Qx)\varphi_2,$$

$$(Qx)\varphi_1 \vee (Qx)\varphi_2 \models (Qx)(\varphi_1 \vee \varphi_2).$$

(3) *If* Q *is a possibility quantifier, then for any* $\varphi_1, \varphi_2 \in \mathrm{Wff}$, *we have*:

$$(Qx)\big(\varphi_1(x) \vee \varphi_2(x)\big) \equiv (Qx)\varphi_1(x) \vee (Qx)\varphi_2(x).$$

Proof. (1) The "only if" part is a direct corollary of Lemma 12(2). For the "if" part, let I be an interpretation and X be the domain of I. We want to show that $T_I(\varphi_1) \leqslant T_I(\varphi_2)$. We set $Q_X = \Pi_{\pi_{x^I}}$, where x^I is the assignment of I to x, and $\Pi_{\pi_{x^I}}$ is the fuzzy measure induced by the singleton possibility distribution π_{x^I} (see Example 4(1)). Then from $(Qx)\varphi_1 \models (Qx)\varphi_2$ and a simple calculation we know that

$$T_I(\varphi_1) = \int T_{I\{./x\}}(\varphi_1) \circ \Pi_{\pi_{x^I}} = T_I\big((Qx)\varphi_1\big) \leqslant T_I\big((Qx)\varphi_2\big) = \int T_{I\{./x\}}(\varphi_2) \circ \Pi_{\pi_{x^I}} = T_I(\varphi_2).$$

(2) is immediate from (1).

(3) For any interpretation I, let X be the domain of I, then it follows that

$$\begin{aligned}
&T_I\big((Qx)(\varphi_1 \vee \varphi_2)\big)\\
&\quad= \sup_{\lambda\in[0,1]} \min\big[\lambda, Q_X\big(\{u \in X\colon \max\big(T_{I\{u/x\}}(\varphi_1), T_{I\{u/x\}}(\varphi_2)\big) \geqslant \lambda\}\big)\big]\\
&\quad= \sup_{\lambda\in[0,1]} \min\big[\lambda, Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi_1) \geqslant \lambda\} \cup \{u \in X : T_{I\{u/x\}}(\varphi_2) \geqslant \lambda\}\big)\big]\\
&\quad= \sup_{\lambda\in[0,1]} \min\big[\lambda, \max\big(Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi_1) \geqslant \lambda\}\big), Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi_2) \geqslant \lambda\}\big)\big)\big]\\
&\quad= \sup_{\lambda\in[0,1]} \max\big\{\min\big[\lambda, Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi_1) \geqslant \lambda\}\big)\big], \min\big[\lambda, Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi_2) \geqslant \lambda\}\big)\big]\big\}\\
&\quad= \max\big\{\sup_{\lambda\in[0,1]} \min\big[\lambda, Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi_1) \geqslant \lambda\}\big)\big], \sup_{\lambda\in[0,1]} \min\big[\lambda, Q_X\big(\{u \in X\colon T_{I\{u/x\}}(\varphi_2) \geqslant \lambda\}\big)\big]\big\}\\
&\quad= T_I\big((Qx)\varphi_1 \vee (Qx)\varphi_2\big). \qquad \square
\end{aligned}$$

Second, we see how the consequence relation between two quantified propositions depends on the strength of involved quantifiers.

Proposition 30. *Let Q_1 and Q_2 be two quantifiers. Then $Q_1 \sqsubseteq Q_2$ if and only if for any $\varphi \in \mathrm{Wff}$, $(Q_1x)\varphi \models (Q_2x)\varphi$ always holds.*

Proof. The "only if" part is obvious. For the "if" part, we only need to show that for any set X and for any $E \subseteq X$, $Q_{1X}(E) \leqslant Q_{2X}(E)$. We can find some $P \in \mathbf{F}_n$ with $n > 0$ because it is assumed that $\bigcup_{n=1}^{\infty} \mathbf{F}_n \neq \emptyset$. Let $\varphi = P(x, \ldots, x)$ and consider the interpretation I in which the domain is X and

$$P^I(u_1, \ldots, u_n) = \begin{cases} 1, & \text{if } u_1 = \cdots = u_n \in E,\\ 0, & \text{otherwise.}\end{cases}$$

Then

$$T_I\big((Q_1x)\varphi\big) = \sup_{F\subseteq X} \min\big[\inf_{u\in F} P^I(u, \ldots, u), Q_X(F)\big] = \sup_{F\subseteq E} Q_X(F) = Q_{1X}(E)$$

because

$$\inf_{u\in F} P^I(u, \ldots, u) = \begin{cases} 1, & \text{if } F \subseteq E,\\ 0, & \text{otherwise.}\end{cases}$$

A similar calculation leads to $T_I((Q_2x)\varphi) = Q_{2X}(E)$. Since $(Q_1x)\varphi \models (Q_2x)\varphi$, we have $T_I((Q_1x)\varphi) \leqslant T_I((Q_2x)\varphi)$, and this completes the proof. $\square$

Third, we show that a proposition with the meet or union of two quantifiers can be transformed to the conjunction or disjunction, respectively, of the propositions with component quantifiers.

Proposition 31. *For any quantifiers Q_1, Q_2, individual variable x and $\varphi \in \mathrm{Wff}$, we have*:

(1) $((Q_1 \sqcap Q_2)x)\varphi \equiv (Q_1x)\varphi \wedge (Q_2x)\varphi$;
(2) $((Q_1 \sqcup Q_2)x)\varphi \equiv (Q_1x)\varphi \vee (Q_2x)\varphi$.

Proof. (1) For any interpretation I with domain X, it is clear that

$$T_I\big(\big((Q_1 \sqcap Q_2)x\big)\varphi\big) \leqslant T_I\big((Q_1x)\varphi \wedge (Q_2x)\varphi\big).$$

Conversely, we have

$$
\begin{aligned}
T_I\big((Q_1x)\varphi \wedge (Q_2x)\varphi\big) &= \min\big[T_I\big((Q_1x)\varphi\big), T_I\big((Q_2x)\varphi\big)\big] \\
&= \min\Big[\sup_{F_1\subseteq X} \min\Big(\inf_{u\in F_1} T_{I\{u/x\}}(\varphi), Q_{1X}(F_1)\Big), \sup_{F_2\subseteq X} \min\Big(\inf_{u\in F_2} T_{I\{u/x\}}(\varphi), Q_{2X}(F_2)\Big)\Big] \\
&= \sup_{F_1,F_2\subseteq X} \min\Big(\inf_{u\in F_1} T_{I\{u/x\}}(\varphi), \inf_{u\in F_2} T_{I\{u/x\}}(\varphi), Q_{1X}(F_1), Q_{2X}(F_2)\Big) \\
&= \sup_{F_1,F_2\subseteq X} \min\Big(\inf_{u\in F_1\cup F_2} T_{I\{u/x\}}(\varphi), Q_{1X}(F_1), Q_{2X}(F_12)\Big) \\
&\leqslant \sup_{F_1,F_2\subseteq X} \min\Big(\inf_{u\in F_1\cup F_2} T_{I\{u/x\}}(\varphi), Q_{1X}(F_1\cup F_2), Q_{2X}(F_1\cup F_2)\Big) \\
&\leqslant \sup_{F\subseteq X} \min\Big(\inf_{u\in F} T_{I\{u/x\}}(\varphi), Q_{1X}(F), Q_{2X}(F)\Big) \\
&= \sup_{F\subseteq X} \min\Big(\inf_{u\in F} T_{I\{u/x\}}(\varphi), (Q_1\sqcap Q_2)_X(F)\Big) \\
&= T_I\big(\big((Q_1\sqcap Q_2)x\big)\varphi\big).
\end{aligned}
$$

(2) Let

$$F_{X,\lambda} = \big\{u\in X\colon\ T_{I\{u/x\}}(\varphi) \geqslant \lambda\big\}.$$

Then it follows that

$$
\begin{aligned}
T_I\big(\big((Q_1\sqcup Q_2)x\big)\varphi\big) &= \sup_{\lambda} \min\big[\lambda, (Q_1\sqcup Q_2)_X(F_{X,\lambda})\big] \\
&= \sup_{\lambda} \min\big[\lambda, \max\big(Q_{1X}(F_{X,\lambda}), Q_{2X}(F_{X,\lambda})\big)\big] \\
&= \sup_{\lambda} \max\big[\min\big(\lambda, Q_{1X}(F_{X,\lambda})\big), \min\big(\lambda, Q_{2X}(F_{X,\lambda})\big)\big] \\
&= \max\big[\sup_{\lambda} \min\big(\lambda, Q_{1X}(F_{X,\lambda})\big), \sup_{\lambda} \min\big(\lambda, Q_{2X}(F_{X,\lambda})\big)\big] \\
&= \max\big(T_I\big((Q_1x)\varphi\big), T_I\big((Q_2x)\varphi\big)\big) \\
&= T_I\big((Q_1x)\varphi \vee (Q_2x)\varphi\big). \qquad \square
\end{aligned}
$$

We pointed out in the remark after Definition 18 that more general meet and union operations of quantifiers may be defined by using a t-norm and t-conorm in the places of "min" and "max", respectively. The above proposition can be generalized to such general meet and union of quantifiers. To this end, we have to consider a generalization of Sugeno's integral, which is obtained by replacing "min" in Definition 9 with a t-norm (see [38] for a detailed discuss on such generalized Sugeno's integrals).

An inference scheme with linguistic quantifiers discussed in [25, Section 7] is as follows:

$$\frac{(Q_1x)\varphi_1, \ldots, (Q_nx)\varphi_n}{(Qx)\varphi =?}.$$

The above proposition enables us to give a solution to this inference problem. Indeed, from the above proposition we know that the following inference is valid:

$$\frac{(Q_1x)\varphi_1, \ldots, (Q_nx)\varphi_n}{\big((Q_1\sqcap\cdots\sqcap Q_n)x\big)(\varphi_1\vee\cdots\vee\varphi_n)}.$$

However, it should be pointed out that such a solution is usually not optimal; more precisely, there may be some logical formula ψ of the form $(Qx)\varphi$ such that

$$\models \psi \to \big((Q_1\sqcap\cdots\sqcap Q_n)x\big)(\varphi_1\vee\cdots\vee\varphi_n),$$

but

$$\models \big((Q_1\sqcap\cdots\sqcap Q_n)x\big)(\varphi_1\vee\cdots\vee\varphi_n) \to \psi$$

does not hold.

The next proposition indicates that quantifiers do not shed any influence on bound variables.

Proposition 32. *For any quantifier Q and for any $\varphi, \psi \in \mathrm{Wff}$, if individual variable x is not free in ψ, then we have*:

$$(Qx)\varphi \wedge \psi \equiv (Qx)(\varphi \wedge \psi);$$
$$(Qx)\varphi \vee \psi \equiv (Qx)(\varphi \vee \psi).$$

Proof. We only demonstrate the first equivalence relation, and the second is similar. For any interpretation I, it holds that

$$T_I\big((Qx)(\varphi \wedge \psi)\big) = \int T_{I,x}(\varphi \wedge \psi) \circ Q_X = \int \min\big[T_{I\{u/x\}}(\varphi), T_{I\{u/x\}}(\psi)\big] \circ Q_X,$$

where X is the domain of I. Since x is not free in ψ, it is easy to see that $T_{I\{u/x\}}(\psi) = T_I(\psi)$ for every $u \in X$. With Lemma 12(3), we obtain

$$T_I\big((Qx)(\varphi \wedge \psi)\big) = \min\left[\int T_{I\{u/x\}}(\varphi) \circ Q_X, T_I(\psi)\right] = \min\big[T_I\big((Qx)\varphi\big), T_I(\psi)\big] = T_I\big((Qx)\varphi \wedge \psi\big). \qquad \square$$

To conclude this section, we observe the function of dual quantifiers. It is worth noting that the fuzzy validity $\models^{Fuz}$ in the following proposition cannot be strengthened by the stricter validity $\models$.

Proposition 33. *For any quantifier Q and for any $\varphi \in \mathrm{Wff}$, it holds that*

$$\models^{Fuz} \sim (Qx)\varphi \leftrightarrow (Q^*x) \sim \varphi,$$

where Q^* is the dual of Q.

Proof. We set

$$\psi_1 = \sim (Qx)\varphi \rightarrow (Q^*x) \sim \varphi$$

and

$$\psi_2 = (Q^*x) \sim \varphi \rightarrow \sim (Qx)\varphi.$$

Then for each interpretation I with domain X, we have

$$T_I\big(\sim (Qx)\varphi \leftrightarrow (Q^*x) \sim \varphi\big) = \min\big[T_I(\psi_1), T_I(\psi_2)\big],$$

and it suffices to show that $T_I(\psi_1) \geqslant \frac{1}{2}$ and $T_I(\psi_2) \geqslant \frac{1}{2}$. Furthermore, it holds that

$$\begin{aligned} T_I(\psi_1) &= \max\big[T_I\big((Qx)\varphi\big), T_I(Q^*x) \sim \varphi\big)\big] \\ &= \max\left[\int T_{I\{u/x\}}(\varphi) \circ Q_X, \int \big(1 - T_{I\{u/x\}}(\varphi)\big) \circ Q_X^*\right]. \end{aligned}$$

If

$$\int T_{I\{u/x\}}(\varphi) \circ Q_X \geqslant \frac{1}{2},$$

then it is obvious that $T_I(\psi_1) \geqslant \frac{1}{2}$. Otherwise, with Lemma 10 we obtain

$$\sup_{F \subseteq X} \min\big[\inf_{x \in F} T_{I\{u/x\}}(\varphi), Q_X(F)\big] = \int T_{I\{u/x\}}(\varphi) \circ Q_X < \frac{1}{2}.$$

For any $\varepsilon > 0$, let $F(\varepsilon) = \{u \in X\colon\ T_{I\{u/x\}}(\varphi) < \frac{1}{2}\}$. Then for all $u \in F(\varepsilon)$, $1 - T_{I\{u/x\}}(\varphi) \geqslant \frac{1}{2} - \varepsilon$, and

$$\inf_{u \in F(\varepsilon)} \big(1 - T_{I\{u/x\}}(\varphi)\big) \geqslant \frac{1}{2} - \varepsilon.$$

On the other hand, since $X - F(\varepsilon) = \{u \in X\colon\ T_{I\{u/x\}}(\varphi) \geqslant \frac{1}{2} + \varepsilon\}$, it follows that

$$\inf_{u \in X - F(\varepsilon)} T_{I\{u/x\}}(\varphi) \geqslant \frac{1}{2} + \varepsilon.$$

Note that

$$\min\Big[\inf_{x\in X-F(\varepsilon)} T_{I\{u/x\}}(\varphi),\, Q_X\big(X-F(\varepsilon)\big)\Big] \leqslant \int T_{I\{u/x\}}(\varphi)\circ Q_X < \frac{1}{2}.$$

We know that $Q_X(X-F(\varepsilon)) < \frac{1}{2}$. This yields $Q_X^*(F(\varepsilon)) = 1 - Q_X(X-F(\varepsilon)) > \frac{1}{2}$ and

$$\int \big(1-T_{I\{u/x\}}(\varphi)\big)\circ Q_X^* \geqslant \min\Big[\inf_{u\in F(\varepsilon)}\big(1-T_{I\{u/x\}}(\varphi)\big),\, Q_X^*(F(\varepsilon))\Big] \geqslant \frac{1}{2}-\varepsilon.$$

Let $\varepsilon\to 0$. Then it holds that

$$\int \big(1-T_{I\{u/x\}}(\varphi)\big)\circ Q_X^* \geqslant \frac{1}{2}$$

and $T_I(\psi_1)\geqslant\frac{1}{2}$.

We now turn to consider ψ_2. It is clear that

$$\begin{aligned} T_I(\psi_2) &= \max\left[1-\int \big(1-T_{I\{u/x\}}(\varphi)\big)\circ Q_X^*,\, 1-\int T_{I\{u/x\}}(\varphi)\circ Q_X\right] \\ &= 1-\min\left[\int \big(1-T_{I\{u/x\}}(\varphi)\big)\circ Q_X^*,\, \int T_{I\{u/x\}}(\varphi)\circ Q_X\right]. \end{aligned}$$

To show that $T_I(\psi_2)\geqslant\frac{1}{2}$, it suffices to prove

$$\min\left[\int \big(1-T_{I\{u/x\}}(\varphi)\big)\circ Q_X^*,\, \int T_{I\{u/x\}}(\varphi)\circ Q_X\right] \leqslant \frac{1}{2}.$$

If

$$\int T_{I\{u/x\}}(\varphi)\circ Q_X \leqslant \frac{1}{2},$$

we are done. Otherwise, we have

$$\sup_{F\subseteq X}\min\Big[\inf_{u\in F} T_{I\{u/x\}}(\varphi),\, Q_X(F)\Big] = \int T_{I\{u/x\}}(\varphi)\circ Q_X > \frac{1}{2}.$$

Therefore, there exists $F_0\subseteq X$ such that

$$\inf_{u\in F_0} T_{I\{u/x\}}(\varphi) > \frac{1}{2}$$

and $Q_X(F_0)>\frac{1}{2}$. Now, for any $u\in F_0$, $T_{I\{u/x\}}(\varphi)>\frac{1}{2}$, $1-T_{I\{u/x\}}(\varphi)<\frac{1}{2}$, and $Q_X^*(X-F_0)=1-Q_X(F_0)<\frac{1}{2}$. We are going to show that

$$\int \big(1-T_{I\{u/x\}}(\varphi)\big)\circ Q_X^* = \sup_{F\subseteq X}\min\Big[\inf_{u\in F}\big(1-T_{I\{u/x\}}(\varphi)\big),\, Q_X^*(F)\Big] \leqslant \frac{1}{2}.$$

To this end, we only need to demonstrate that for each $F\subseteq X$, $Q_X^*(F)\leqslant\frac{1}{2}$ or

$$\inf_{u\in F}\big(1-T_{I\{u/x\}}(\varphi)\big) \leqslant \frac{1}{2}.$$

In fact, if $F\subseteq X-F_0$, then $Q_X^*(F)\leqslant Q_X^*(X-F_0)<\frac{1}{2}$, and if $F\not\subseteq X-F_0$, then there exists $u_0\in F\cap F_0$ and

$$\inf_{u\in F}\big(1-T_{I\{u/x\}}(\varphi)\big) \leqslant 1-T_{I\{u_0/x\}}(\varphi) < \frac{1}{2}.$$

This completes the proof. □

Combining the above results we obtain a prenex normal form theorem for logical formulas with linguistic quantifiers.

Corollary 34 *(Prenex Normal Form). For any* $\varphi \in \mathrm{Wff}$*, there exists* $\psi \in \mathrm{Wff}$ *satisfying the following two conditions*:

(i) ψ *is in the form of*

$$(Q_1 y_1) \ldots (Q_n y_n) M,$$

where $n \geqslant 0$*,* $Q_1, \ldots, Q_n$ *are quantifiers, and* $M \in \mathrm{Wff}$ *does not contain any quantifier*; *and*

(ii) $\models^{Fuz} \varphi \leftrightarrow \psi$.

Proof. Immediate from Propositions 31–33. □

6. Cardinal quantifiers

The studies on generalized quantifiers has traditionally been based on cardinalities [2,29,36]. The reason is that when a quantified proposition is considered, what really concerns us is how many individuals satisfies the proposition, and it is usually irrelevant to know what are they concretely. Consequently, most methods of evaluating linguistic quantification in the previous literature are based on cardinalities of fuzzy sets too. Our fuzzy measure and Sugeno integral approach to linguistic quantifiers is in a sense presented in a wider framework in which no cardinality condition is required. This section aims at clarifying the relationship between the semantics proposed in Section 4 and the cardinality-based semantics of linguistic quantifiers. To this end, we have to introduce a special class of quantifiers.

Definition 35. A quantifier Q is called a cardinal quantifier if for any two nonempty sets X and X', for any bijection $f : X \to X'$, and for any $E \in \wp_X$ with $f(E) \in \wp'_{X'}$,

$$Q_X(E) = Q_{X'}\big(f(E)\big),$$

where $f(E) = \{f(x)\colon\ x \in E\}$.

In the above definition, the requirement that $f : X \to X'$ is a bijection implies that X and X' has the same cardinality, and E and $f(E)$ also has the same cardinality. Consider the following two quantified statements with the same quantifier Q: $\varphi =$ "Q *Xs are As*" and $\varphi' =$ "Q *X's are A's*". Suppose E is the set of elements in X that satisfy the property A, that is, $E = \{x \in X\colon\ x \text{ satisfies } A\}$. Similarly, let $E' = \{x \in X'\colon\ x \text{ satisfies } A'\}$. Then the condition in the above definition means that φ and φ' are equivalent provided X and X' has the same cardinality and so do A and A', no matter what are elements of X, X', A and A'.

The notion of generalized quantifier introduced in [45] is similar to cardinal quantifier with the only difference that the set of truth values was allowed to be any complete lattice in [45].

The notion of cardinal quantifier has an equivalent and simplified definition given in terms of cardinal numbers. For any cardinal number α, we write $m(\alpha)$ for the set of all (increasing) functions

$$f : \{\text{cardinal number } \beta\colon\ \beta \leqslant \alpha\} \to [0, 1]$$

such that $\beta_1 \leqslant \beta_2$ implies $f(\beta_1) \leqslant f(\beta_2)$.

Definition 36. A numeric quantifier is a choice function

$$q : \alpha \mapsto q_\alpha \in m(\alpha)$$

of the class $\{m(\alpha)\colon\ \alpha \text{ is a cardinal number}\}$.

For any cardinal quantifier Q, we define:

$$|Q| : \alpha \mapsto |Q|_\alpha \quad \text{for any cardinal number } \alpha,$$

$$|Q|_\alpha(\beta) \stackrel{\text{def}}{=} Q_X(E) \quad \text{for any } \beta \leqslant \alpha,$$

where X is some set with $|X| = \alpha$ and E is some subset of X with $|E| = \beta$. From Definition 35 we know that Q is well-defined, and it is a numeric quantifier. Thus, cardinal quantifier Q is represented by a numeric quantifier $|Q|$. Conversely, for each numeric quantifier q, we put

$$[q]_X(E) \stackrel{\text{def}}{=} q_{|X|}\big(|E|\big)$$

for any nonempty set X, and for any $E \subseteq X$. It is easy to see that $[q]$ is a cardinal quantifier, and thus numeric quantifier q induces a cardinal quantifier $[q]$.

Example 37. The universal quantifier ($\forall$) is not a cardinal quantifier because for an infinite set X, it is possible that X has a proper subset E with $|E| = |X|$. The existential quantifier ($\exists$) is a cardinal quantifier, and for any cardinal numbers α, β with $\alpha \leqslant \beta$,

$$|\exists|_\alpha(\beta) = \begin{cases} 1, & \text{if } \beta > 0, \\ 0, & \text{otherwise.} \end{cases}$$

The above example shows that the universal quantifier cannot be treated by the usual approach based on cardinality when the discourse universe is infinite. This exposes indeed one of the advantages of our fuzzy measure approach to linguistic quantifiers.

In order to give another way of evaluating the truth value of a proposition with cardinal quantifier and to compare fuzzy measure semantics of linguistic quantifiers with others, we need the notion of consistency between two fuzzy sets.

Definition 38. Let X be a nonempty set, and let A, B be two fuzzy subsets of X. Then the consistency index of A and B is defined as

$$Con(A, B) = \sup_{x \in X} \min\big[A(x), B(x)\big].$$

The notion of consistency was originally introduced by Zadeh in [55], and it have also been used by other authors to serve as similarity measures between two fuzzy sets [58]. Moreover, the quantity $1 - Con(A, \overline{B})$ was introduced as inclusion grade of fuzzy set A in fuzzy set B (see [11, page 25]), where $\overline{B}$ is the complement of B, that is, $\overline{B}(x) = 1 - B(x)$ for each $x \in X$.

A concept of fuzzy cardinality of fuzzy set is also needed.

Definition 39. Let X be a nonempty set, and let A be a fuzzy subset of X. Then the fuzzy cardinality of A is defined to be the fuzzy subset $FC(A)$ of {cardinal number α: $\alpha \leqslant |X|$} with

$$FC(A)(\alpha) = \sup\big\{\lambda \in [0, 1]\colon\ |A_\lambda| = \alpha\big\}$$

for any $\alpha \leqslant |X|$, where $A_\lambda = \{x \in X\colon\ A(x) \geqslant \lambda\}$ is the λ-cut of A.

We give a simple example to illustrate the notion of fuzzy cardinality.

Example 40. We adopt the Zadeh's notation for fuzzy sets, that is, a fuzzy subset A of a set X is denoted by

$$A = \sum_{x \in X} A(x)/x.$$

Let $X = \{a_1, a_2, \ldots, a_{10}\}$. We consider the following fuzzy subset A of X:

$$A = 0.9/a_1 + 0.4/a_2 + 0.7/a_4 + 0.1/a_5 + 0.82/a_6 + 0.83/a_7 + 0.9/a_8 + 1/a_9 + 1/a_{10}.$$

Then a routine calculation yields

$$FC(A) = 1/2 + 0.9/4 + 0.83/5 + 0.82/6 + 0.7/7 + 0.4/8 + 0.1/9.$$

It is easy to see that if A is a crisp set and $|A| = \alpha$ then $FC(A) = 1/\alpha$ (in Zadeh's notation). Conversely, if X is a finite set, A is a fuzzy subset of X and $FC(A) = 1/\alpha$, then A is a crisp set and $|A| = \alpha$. This indicates that $FC(\cdot)$ is a reasonable fuzzy generalization of the concept of cardinality for crisp sets.

In the literature, there are mainly two ways of defining cardinalities of fuzzy sets. The first gives a (non-fuzzy) real number as the cardinality of a fuzzy set A; for example, if A is a fuzzy subset of a finite set X, then its sigma-count is defined by

$$\Sigma Count(A) = \sum_{x \in X} A(x)$$

(see [56]). In the second way, the cardinality of a fuzzy set A is defined to be a fuzzy subsets of nonnegative integers, and typical examples are Zadeh's $FGCount(\cdot)$, $FLCount(\cdot)$ and $FECount(\cdot)$ [56] and Ralescu's $card(\cdot)$ [31]. Obviously, $FC(\cdot)$ in Definition 39 is also given in the second way, but here we are concerned not only finite sets and infinite cardinalities are allowed too. Moreover, even for the case of finite cardinalities, it is easy to observe that $FC(\cdot)$ is different from $FGCount(\cdot)$, $FLCount(\cdot)$, $FECount(\cdot)$ and $card(\cdot)$. We are not going to give a careful comparison of them.

Now we are able to present the main result of this section. It clarifies the relation between our Sugeno integral semantics of linguistic quantifiers and the usual approach based on cardinality.

Proposition 41. *If Q is a cardinal quantifier, then for any $\varphi \in$ Wff and for any interpretation I, we have*

$$T_I\big((Qx)\varphi\big) = Con\big(FC\big(T_I(\varphi)\big), |Q|_{|X|}\big),$$

where $T_I(\varphi)$ is considered as a fuzzy subset of X with $T_I(\varphi)(u) = T_{I\{u/x\}}(\varphi)$ for all $u \in X$, and $|Q|$ is the numeric quantifier induced by Q.

Proof.

$$\begin{aligned}
T_I\big((Qx)\varphi\big) &= \sup_{\lambda\in[0,1]} \min\big[\lambda, Q_X\big(\big\{u \in X\colon\ T_{I\{u/x\}}(\varphi) \geqslant \lambda\big\}\big)\big] \\
&= \sup_{\lambda\in[0,1]} \min\big[\lambda, |Q|_{|X|}\big(\big|\big\{u \in X\colon\ T_{I\{u/x\}}(\varphi) \geqslant \lambda\big\}\big|\big)\big] \\
&= \sup_{\lambda\in[0,1]} \min\big[\lambda, |Q|_{|X|}\big(\big|\big(T_I(\varphi)\big)_\lambda\big|\big)\big] \\
&= \sup_{\alpha\leqslant|X|} \sup_{\lambda\in[0,1]\ \text{s.t}\ |(T_I(\varphi))_\lambda|=\alpha} \min\big[\lambda, |Q|_{|X|}(\alpha)\big] \\
&= \sup_{\alpha\leqslant|X|} \min\Big[\sup_{\lambda\in[0,1]\ \text{s.t}\ |(T_I(\varphi))_\lambda|=\alpha} \lambda, |Q|_{|X|}(\alpha)\Big] \\
&= \sup_{\lambda\leqslant|X|} \min\big[FC\big(T_I(\varphi)\big)(\alpha), |Q|_{|X|}(\alpha)\big] \\
&= Con\big(FC\big(T_I(\varphi)\big), |Q|_{|X|}\big). \qquad \square
\end{aligned}$$

7. Some simple applications

To illustrate the utility of the Sugeno integral semantics of linguistic quantifiers, in this section we present three simple examples.

The first example is concerned with the weather in a week.

Example 42. Recall from Example 16 that the quantifier $Q =$ "many" is defined on a finite set X as follows:

$$Q_X(E) = \frac{|E|}{|X|}$$

for each $E \subseteq X$, where $|E|$ is the cardinality of E. Let $P_1 =$ "to be cloudy" and $P_2 =$ "to be cold" be two (fuzzy) linguistic predicates. Consider the interpretation I in which the domain is

$X =$ {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}

Table 1
Truth table of linguistic predicates P_1 and P_2

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
P_1^I	0.1	0	0.5	0.8	0.6	1	0.2
P_2^I	1	0.9	0.4	0.7	0.3	0.4	0

Table 2
Health condition of 10 students

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
$H(x)$	0.73	0.1	0.95	1	0.84	0.67	0.7	0.9	1	0.81

and the truth values of P_1 and P_2 are given in Table 1. Then the logical formula

$$\varphi = (Qx)\big(P_1(x) \wedge \sim P_2(x)\big)$$

means that "many days (in this week) are cloudy but not cold", and its truth value under I is

$$T_I(\varphi) = \max_{E \subseteq X}\big[\min_{u \in E} \min\big(P_1^I(u), 1 - P_2^I(u)\big)\big] = \frac{3}{7} \approx 0.43.$$

The next example demonstrates applicability of Proposition 27 in health data summarization.

Example 43. Consider a set $X = \{s_1, s_2, \ldots, s_{10}\}$ consisting of 10 students. The respective health condition of these students is indicated by Table 2, where the symbol H stands for the linguistic predicate "to be healthy". We want to find a suitable linguistic quantifier from $Q_1 =$ "some", $Q_2 =$ "at least three", $Q_3 =$ "many", $Q_4 =$ "most", $Q_5 =$ "almost all" and $Q_6 =$ "all" to summarize the above data (see Examples 15 and 16 for the definitions of these quantifiers). In other words, we hope to have a high level description of the whole health condition of this group of students. It is required that such a summarization should hold with a high truth value, say, $\geqslant 0.7$. From the above table, we have

$$E \stackrel{\text{def}}{=} \big\{x \in X\colon\ H(x) \geqslant 0.7\big\} = \{s_1, s_3, s_4, s_5, s_7, s_8, s_9, s_{10}\}$$

and

$$Q_{4X}(E) = \left(\frac{|E|}{|X|}\right)^{3/2} = \left(\frac{4}{5}\right)^{3/2} > 0.7 > \left(\frac{4}{5}\right)^2 = Q_{5X}(E).$$

With Proposition 27 we know that

$$T_I\big((Q_4x)H(x)\big) > 0.7 > T_I\big((Q_5x)H(x)\big),$$

where I is the interpretation given according Table 2. So, what we can say is "most students are healthy".

We finally consider the problem of soft database queries. This kind of problems were handled in [23] by employing Zadeh's approach [56] to linguistic quantifiers. However, here we adopt the Sugeno integral semantics developed in the previous sections.

Example 44. Suppose that a record or entity is a list of attributes, a file is a collection of records of the same type, and a database is a collection of files of various types. Formally, a type is a tuple $T = (x_1, x_2, \ldots, x_k)$ of attribute names. For each $i \leqslant k$, we assume that the set of possible values of the attribute x_i is V_i. Then a record of type T can be written as

$$R = (x_1 : a_1, x_2 : a_2, \ldots, x_k : a_k),$$

where $a_i \in V_i$ is the value of attribute x_i in this record for every $i \leqslant k$. On the other hand, a (soft) query of type T may be formulated as follows:

$q =$ "*find all records such that Q of the attributes out of*

$\{x_1$ *is* $F_1,\ x_2$ *is* $F_2,\ \ldots,\ x_k$ *is* $F_k\}$ *match*",

where Q is a linguistic quantifier, and for any $i \leqslant k$, F_i is a given linguistic predicate which represents a (soft) constraint on the attribute x_i and which can be defined as a fuzzy subset of V_i. We usually write the types of record R and query q as $Type(R)$ and $Type(q)$, respectively. The above query can be expressed as a formula in our logical language $\mathbf{L_q}$:

$$\varphi = (Qx)\big(value(x) \in constraint(x)\big),$$

where x is an individual variable, both "*value*" and "*constraint*" are unary functions, and $\in$ is a binary predicate. We can imagine that a record R and a query q give an interpretation I of the formula φ: the discourse universe is the type $T = \{x_1, x_2, \ldots, x_k\}$, and

$$value^I(x_i) = a_i,$$
$$constraint^I(x_i) = F_i,$$
$$\in^I(a_i, F_i) = F_i(a_i)$$

for any $i \leqslant k$, for any $a_i \in V_i$ and for any fuzzy subset F_i of V_i. Now the degree to which a record R matches the query q is calculated as the truth value of the formula φ:

$$Match(q, R) = T_I(\varphi) = \int h \circ Q_T,$$

where $h(x_i) = F_i(a_i)$ for every $i \leqslant k$. Furthermore, a database can be defined as

$$D = \bigcup_{j=1}^{m} F_j,$$

where F_j is a file of type T_j, that is, a set of some records with type T_j for each $j \leqslant m$. Let λ be a pre-specified threshold. If the matching degree of a record R and the query q exceeds λ, then R is introduced into the output file of q. Therefore, the output file of query q is given as

$$OUTPUT(q) = \big\{R \in D\colon Type(R) = Type(q) \text{ and } Match(q, R) \geqslant \lambda\big\}.$$

Let us now consider a concrete query

$q =$ "*find all records such that almost all of the attributes out of*

$\{x_1$ *is small*, x_2 *is small*, x_3 *is big*, x_4 *is around* 4, x_5 *is big*, x_6 *is very big*,

x_7 *is small*, x_8 *is small*, x_9 *is very small*, x_{10} *is very big*$\}$ *match*",

where the type of q is $T = \{x_1, x_2, \ldots, x_{10}\}$. Assume that $V_1 = V_2 = \cdots = V_{10} = \{1, 2, \ldots, 8\}$ and

"*small*" $= 1/1 + 0.8/2 + 0.3/3$,

"*very small*" $= 1/1 + 0.64/2 + 0.09/3$,

"*big*" $= 0.3/6 + 0.8/7 + 1/8$,

"*very big*" $= 0.09/6 + 0.64/7 + 1/8$,

"*around* 4" $= 0.3/2 + 08/3 + 1/4 + 0.8/5 + 0.3/6$.

Furthermore, we consider the following record

$$R = (x_1 : 7, x_2 : 2, x_3 : 8, x_4 : 3, x_5 : 6, x_6 : 6, x_7 : 1, x_8 : 2, x_9 : 2, x_{10} : 7).$$

The interpretation I determined by the record R and the query q is then given by Table 3.

Using Lemma 11 it is routine to compute the matching degree of the record R to the query q:

$$Match(q, R) = \int h \circ almost\ all_T = \min\big[0.64, almost\ all_T\big(\{x_2, x_3, x_4, x_7, x_8, x_9, x_{10}\}\big)\big] = 0.49,$$

where the quantifier "almost all" is defined according to Example 16.

Table 3
The interpretation I determined by record R and query q

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
$h(x_i) = F_i(a_i)$	0	0.8	1	0.8	0.3	0.09	1	0.8	0.64	0.64

8. Discussion

The main aim of this section is to clarify further the relationship between the results obtained in this paper and related works.

Zadeh [56] distinguished two kinds of fuzzy quantifiers, absolute quantifiers and relative quantifiers. An absolute quantifier refers to a (fuzzily defined) number such as "at least three", "few" and "not very many", while a relative quantifier refers to a proportion such as "most" and "at most half". Then truth evaluation of quantified statements is performed by calculating (absolute and relative) cardinalities of fuzzy sets. In this paper, however, evaluation of quantification is directly done over subsets (not their cardinalities) of the discourse universe. This is different from the way considered in [56] and enables us not need to distinguish absolute and relative quantifiers.

In [25], a linguistic quantifier is defined as a constraint on probability values, and it can be viewed as an alternative form of imprecise probability. From the angle of mathematical representation, linguistic quantifiers dealt with in [25] can be identified with the relative quantifiers in [56], both being defined to be fuzzy subsets of the unit interval. More essentially, however, interpretations of quantifiers and reasoning mechanics with quantifiers in [25,56] are quite different: in [25], a quantifier is treated in a probabilistic way and using the voting model, and reasoning with quantifiers is conducted by Bayesian method.

The idea of representing linguistic quantifiers by fuzzy measures and Sugeno's integrals was originally introduced by the author in [42], where some basic logical properties of linguistic quantifiers were presented, including a version of Proposition 33. This paper is a continuation of [42], and indeed some part of the present paper can be seen as an elaboration of [42]. In 1994, Bosc and Lietard [4–7] proposed independently the idea of evaluating linguistically quantified propositions with Sugeno's integrals. However, the motivations of [42] and [4–7] are quite different: the main aim of [42] is to establish a first order logic with fuzzy quantifiers, whereas in [4–7], a close connection between Prade and Yager's approaches to the evaluation of fuzzy quantified statements and Sugeno and Choquet's integrals [8] was found. The different motivations leads to some big differences between [42] (and this paper) and [4–7]. First, the discourse universes considered in [4–7] are required to be finite. Indeed, in almost all previous approaches [1,4–7,9,10,12,15–18,20–23,25,27,28,30–33,39–41,56] to fuzzy quantification, only finite discourse universes are considered because they are mainly application-oriented and usually in real applications finite universes are enough. In the current paper, however, infinite discourse universes are allowed. This enables us to have some deeper logical properties of fuzzy quantifiers. Second, in [4–7] a fuzzy (absolute) quantifier Q over the universe X is still defined to be a fuzzy sets of integers according to Zadeh [56]. Then a fuzzy measure m_Q on X is induced by

$$m_Q(E) = Q\big(|E|\big)$$

for each $E \subseteq X$, where $|E|$ is the cardinality of E, and Sugeno's integrals with respect to m_Q are used to evaluate propositions quantified by Q. (In fact, the process that m_Q is induced from Q is an inverse of the process after Definition 36 that a numeric quantifier is induced from a cardinal quantifier Q.) Thus, the fuzzy quantifiers dealt with in [4–7] are all cardinal in the sense of Definition 36.

Usually, two types of fuzzy quantified statements are identified, namely, type I proposition of the form "Q *Xs are As*" and type II statement of the form "Q *Ds are As*", where X is the discourse universe, Q is a fuzzy quantifier, and A and D are two fuzzy predicates over X [27,28,56]. In this paper, for an elegant logical theory of fuzzy quantifiers, we only consider type I propositions, and each type II proposition may be translated to a type I proposition in the standard ways:

"Q *Ds are As*" $\Leftrightarrow$ "Q *Xs* (*Xs* are *Ds implies Xs* are *As*)"

or

"Q *Ds are As*" $\Leftrightarrow$ "Q *Xs* are *Ds and As*".

The choice of translation methods is determined by the strength of quantifier Q. In general, for stronger quantifiers such as the universal quantifier ($\forall$) we adopt the first method, and for weaker quantifiers such as the existential quantifier ($\exists$) we use the second one.

A list of desirable properties for fuzzy quantification was presented in [1,9,15–18]. Here, we briefly examine the relation between the most relevant ones and the results obtained in this paper. *Independence of order in the elements of the referential* [1,15] is similar to the definition of cardinal quantifier (Definition 35). It is obvious that many quantifiers do not enjoy this property. Example 25 shows that the Sugeno integral semantics of quantifiers satisfies the property of *induced operators* [1,15]. Furthermore, Example 26 shows that this semantics also satisfies *coherence with fuzzy logic* and the property of *correct generalization* [1,9,15,32]. Delgado, Sanchez and Vila [9] proposed that quantification evaluation must be not too "strict". Precisely, this criterion requires that for any quantifier Q there is a fuzzy predicate A such that the evaluation value of sentence "Q *Xs are As*" is neither 0 nor 1 provided Q is not a crisp quantifier. It is easy to see that such a criterion is satisfied by the Sugeno integral semantics of quantifiers. Indeed, let Q be a quantifier and X a set such that there exists $E_0 \subseteq X$ with $0 < Q_X(E_0) < 1$. For any unary predicate symbol P, we consider an interpretation I with X as its domain such that

$$P^I(u) = \begin{cases} \lambda, & \text{if } u \in E_0, \\ 0, & \text{otherwise,} \end{cases}$$

where $0 < \lambda < 1$. Then $0 < T_I((Qx)P(x)) = \min(\lambda, Q_X(E_0)) < 1$. Propositions 29(1) and 30 indicates that both *quantifier monotonicity* [1,9,15,32] and *local monotonicity* are valid in the framework presented in this paper. *Convexity* [1,15]. is then a direct corollary of local monotonicity. *Type I to type II transformation* [1,15,32]. is already discussed in the above paragraph. The property of *decomposition* [1] is given in Proposition 31(1). The property of *external negation* [16] cannot be considered in the Sugeno integral semantics of quantifiers because the complement $\overline{Q}$ of a quantifier Q, defined by $\overline{Q_X}(E) = 1 - Q_X(E)$ for any $E \subseteq X$, does not fulfil the monotonicity in Definition 3 and it is not a quantifier. The property of *antonym* [16] is not true in general. Proposition 33 presents a version of *duality* weaker than that required in [16]. In order to make its practical application possible, it was pointed out in [9,32] that the evaluation method of quantification should be efficient in the sense of computational complexity. Suppose that the domain of interpretation I is a finite set and its cardinality is n. At the first glance, we may think that the evaluation of the truth value $T_I((Qx)\varphi)$ of the quantified formula $(Qx)\varphi$ under I will run in exponential time with complexity $O(2^n)$ (cf. Definitions 9 and 23(ii) and Lemma 10). Fortunately, Lemma 11 provides us with an evaluation algorithm of quantification with complexity $O(n \log n)$.

Continuity or smoothness in behavior [1,18] means that a small perturbation in the membership values of fuzzy properties D or A does not give rise to a drastic change of the evaluation of fuzzy quantified proposition. This property warrants insensibility of fuzzy quantification to noise. A similar problem for compositional rule of fuzzy inference was addressed by the author in [48,52] where linguistic quantifiers were not involved. To give such a property, we first need to introduce a notion of distance between fuzzy sets. Let A and B be two fuzzy subsets of X. Then the distance between A and B is defined to be

$$d(A, B) = \sup_{x \in X} \big|A(x) - B(x)\big|.$$

The following proposition exposes the continuity for our evaluation mechanism of fuzzy quantified statements.

Proposition 45. *For quantifier Q, for any $\varphi, \psi \in$ Wff, and for any interpretation I, we have*:

$$\big|T_I\big((Qx)\varphi\big) - T_I\big((Qx)\psi\big)\big| \leqslant d\big(T_I(\varphi), T_I(\psi)\big).$$

Proof. Note that for any $a, b, c, a_i, b_i \in [0, 1]$ ($i \in J$), it holds that

$$\big|\min(a, c) - \min(b, c)\big| \leqslant |a - b|,$$
$$\big|\inf_{i \in J} a_i - \inf_{i \in J} b_i\big| \leqslant \inf_{i \in J} |a_i - b_i|$$

and

$$\big|\sup_{i \in J} a_i - \sup_{i \in J} b_i\big| \leqslant \sup_{i \in J} |a_i - b_i|.$$

Then it follows that

$$\begin{aligned}\big|T_I\big((Qx)\varphi\big) - T_I\big((Qx)\psi\big)\big| &= \big|\sup_{F\subseteq X}\min\big[\inf_{u\in F} T_{I\{u/x\}}(\varphi),\, Q_X(F)\big] - \sup_{F\subseteq X}\min\big[\inf_{u\in F} T_{I\{u/x\}}(\psi),\, Q_X(F)\big]\big| \\ &\leqslant \sup_{F\subseteq X}\sup_{u\in F}\big|T_{I\{u/x\}}(\varphi) - T_{I\{u/x\}}(\psi)\big| \\ &= \sup_{x\in X}\big|T_{I\{u/x\}}(\varphi) - T_{I\{u/x\}}(\psi)\big| \\ &= d\big(T_I(\varphi), T_I(\psi)\big). \qquad \square\end{aligned}$$

The continuity given in the above proposition is with respect to the change of truth values of statements bound by the same quantifier. Indeed, we also have continuity with respect to perturbation of quantifiers. Let Q_1 and Q_2 be two quantifiers and X a nonempty set. The distance of Q_1 and Q_2 on X is defined to be

$$d_X(Q_1, Q_2) = \sup_{E\subseteq X}\big|Q_{1X}(E) - Q_{2X}(E)\big|.$$

It is worth noting that this definition of distance between quantifiers is similar to variation distance between probability measures [19, page 108].

Proposition 46. *For any quantifiers Q_1 and Q_2, for any $\varphi \in$ Wff, and for any interpretation I, we have*:

$$\big|T_I\big((Q_1x)\varphi\big) - T_I\big((Q_2x)\varphi\big)\big| \leqslant d_X(Q_1, Q_2),$$

where X is the domain of I.

Proof. Similar to Proposition 45. $\square$

It should be pointed out that continuity cannot be expressed in our logical language $\mathbf{L_q}$ given in Section 4.

9. Conclusion

In this paper, we present the Sugeno integral semantics of linguistic quantifiers in which a quantifier is represented by a family of fuzzy measures [35] and the truth value of a quantified proposition is computed by using Sugeno's integral [35]. Several elegant logical properties of linguistic quantifiers are derived, including a prenex normal form theorem. This semantics is compared with the usual cardinality-based approaches (see for example [9,12,31,56]) to linguistic quantifiers.

Three simple applications to data summarization and database queries were presented in Section 7. Nevertheless, more applications are anticipated in other fields such as information fusion [21,25], decision making [20,40] and inductive learning [21].

For further theoretical studies, we wish to examine carefully various model theoretic properties of the logic with linguistic quantifiers and want to see whether it is axiomatizable. The mathematical tool used in this paper for aggregating truth values is the Sugeno integral. As mentioned after Definition 18 and Proposition 31, a generalization of Sugeno's integral was proposed in the previous literature by replacing the operation "min" in Definition 9 with a general t-norm, and this kind of generalized Sugeno's integrals can also be used in our semantics of linguistic quantifiers. It seems that quantifications interpreted in terms of Sugeno's integrals with t-norms allow us to aggregate data, information and knowledge in a "softer" and "more flexible" way. So, a careful examination of linguistic quantifiers modeled by generalized Sugeno's integrals would be another interesting topic. In addition, the Choquet integral is widely used as an aggregation operator in economics, game theory and multi-criteria decision making. It is reasonable to expect that the Choquet integral can also be used to evaluate the truth values of linguistically quantified propositions. Indeed, as pointed out in the last section, Bosc and Lietard [4–7] already noticed a connection between Prade and Yager's representation methods of linguistic quantifiers [30,39,41] and the Choquet integral. Thus, a systematic development of the Choquet integral semantics of linguistic quantifiers would also be an interesting topic.

Acknowledgements

The author is very grateful to the anonymous referees for their invaluable comments and suggestions.

References

[1] S. Barro, A.J. Bugarin, P. Carinena, F. Diaz-Hermida, A framework for fuzzy quantification models analysis, IEEE Transactions on Fuzzy Systems 11 (2003) 89–99.
[2] J. Barwise, R. Cooper, Generalized quantifiers and natural language, Linguistics and Philosophy 4 (1981) 159–219.
[3] L. Biacino, G. Gerla, M.S. Ying, Approximate reasoning based on similarity, Mathematical Logic Quarterly 46 (2000).
[4] P. Bosc, L. Lietard, Monotonic quantifications and Sugeno fuzzy integrals, in: Proc. of 5th IPMU Conference, 1994, pp. 1281–1286.
[5] P. Bosc, L. Lietard, Monotonic quantified statements and fuzzy integrals, in: Proc. of NAFIPS/IFIS/NASA '94, the First International Joint Conference of the North American Fuzzy Information Processing Society Biannual Conference, the Industrial Fuzzy Control and Intelligent Systems Conference, and the NASA Joint Technolo, 1994, pp. 8–12.
[6] P. Bosc, L. Lietard, Monotonic quantifications and Sugeno fuzzy integrals, in: B. Bouchon-Meunier, R.R. Yager, L.A. Zadeh (Eds.), Fuzzy Logic and Soft Computing, World Scientific, Singapore, 1995, pp. 337–344.
[7] P. Bosc, L. Lietard, Fuzzy integrals and database flexible querying, in: Proc. of the Fifth IEEE International Conference on Fuzzy Systems, 1996, pp. 100–106.
[8] G. Choquet, Theory of capacities, Annales de l'Institut Fourier 5 (1955) 131–295.
[9] M. Delgado, D. Sanchez, M.A. Vila, Fuzzy cardinality based evaluation of quantified sentences, International Journal of Approximate Reasoning 23 (2000) 23–66.
[10] F. Diaz-Hermida, A. Bugarin, S. Barro, Definition and classification of semi-fuzzy quantifiers for the evaluation of fuzzy quantified sentences, International Journal of Approximate Reasoning 34 (2003) 49–88.
[11] D. Dubois, H. Prade, Fuzzy Sets and Systems: Theory and Applications, Academic Press, New York, 1980.
[12] D. Dubois, H. Prade, Fuzzy cardinality and the modelling of imprecise quantification, Fuzzy Sets and Systems 16 (1985) 199–230.
[13] N. Friedman, J.Y. Halpern, Plausibility measures: a user's guide, in: Proc. 11th Conference on Uncertainty in Artificial Intelligence (UAI'95), 1995, pp. 175–184.
[14] N. Friedman, J.Y. Halpern, Plausibility measures and default reasoning, Journal of ACM 48 (2001) 648–685.
[15] I. Glockner, DFS—An axiomatic approach to fuzzy quantification, Technical Report TR97-06, Univ. Bielefeld, 1997.
[16] I. Glockner, A framework for evaluating approaches to fuzzy quantification, Technical Report TR99-03, Univ. Bielefeld, 1999.
[17] I. Glockner, Evaluation of quantified propositions in generalized models of fuzzy quantification, International Journal of Approximate Reasoning 37 (2004) 93–126.
[18] I. Glockner, A. Knoll, A formal theory of fuzzy natural language quantification and its role in granular computing, in: W. Pedrycz (Ed.), Granular Computing: An Emerging Paradigm, Physica-Verlag, Heidelberg, 2001, pp. 215–256.
[19] J.Y. Halpern, Reasoning about Uncertainty, MIT Press, Cambridge, MA, 2003.
[20] J. Kacprzyk, M. Fedrizzi, H. Nurmi, Group decision making with fuzzy majorities represented by linguistic quantifiers, in: J.L. Verdegay, M. Delgado (Eds.), Approximate Reasoning Tools for Artificial Intelligence, TUV, Rheinland, 1990, pp. 126–145.
[21] J. Kacprzyk, C. Iwanski, Inductive learning from incomplete and imprecise examples, in: B. Bouchon-Meunier, R.R. Yager, L.A. Zadeh (Eds.), Uncertainty in Knowledge Bases, in: Lecture Notes in Computer Science, vol. 521, Springer, Berlin, 1991, pp. 424–430.
[22] J. Kacprzyk, R.R. Yager, Softer optimization and control models via fuzzy linguistic quantifiers, Information Sciences 34 (1984) 157–178.
[23] J. Kacprzyk, A. Ziolkowski, Databases queries with linguistic quantifiers, IEEE Transctions on Systems, Man and Cybernetics 16 (1986) 474–478.
[24] E.L. Keenan, Some properties of natural language quantifiers: Generalized quantifier theory, Linguistics and Philosophy 25 (2002) 627–654.
[25] J. Lawry, A methodology for computing with words, International Journal of Approximate Reasoning 28 (2001) 51–58.
[26] J. Lawry, A framework for linguistic modelling, Artificial Intelligence 155 (2004) 1–39.
[27] Y. Liu, E.E. Kerre, An overview of fuzzy quantifiers (I): Interpretations, Fuzzy Sets and Systems 95 (1998) 1–21.
[28] Y. Liu, E.E. Kerre, An overview of fuzzy quantifiers (II): Reasoning and applications, Fuzzy Sets and Systems 95 (1998) 135–146.
[29] A. Mostowski, On a generalization of quantifiers, Fundamenta Mathematicae 44 (1957) 17–36.
[30] H. Prade, A two-layer fuzzy pattern matching procedure for the evaluation of conditions involving vague quantifiers, Journal of Intelligent Robotic Systems 3 (1990) 93–101.
[31] A.L. Ralescu, Cardinality, quantifiers, and the aggregation of fuzzy criteria, Fuzzy Sets and Systems 69 (1995) 355–365.
[32] D. Sanchez, Adquisicion de Relaciones entre Atributos en Bases de Datos Relacionales, PhD thesis, Univ. de Granada E. T. S. de Ingenieria Informatica, 1999.
[33] D.G. Schwartz, Dynamic reasoning with qualified syllogisms, Artificial Intelligence 93 (1997) 103–167.
[34] B. Schweizer, A. Sklar, Probabilistic Metric Spaces, North-Holland, Amsterdam, 1983.
[35] M. Sugeno, Theory of fuzzy integrals and its applications, PhD thesis, Tokyo Institute of Technology, 1974.
[36] J. van Benthem, Questions about quantifiers, Journal of Symbolic Logic 49 (1984) 443–466.
[37] P. Walley, Measures of uncertainty in expert systems, Artificial Intelligence 83 (1996) 1–58.
[38] S. Weber, $\perp$-decomposable measures and integrals for Archimedean t-conorms $\perp$, Journal of Mathematical Analysis and Applications 101 (1984) 114–138.
[39] R.R. Yager, Quantified propositions in a linguistic logic, International Journal of Man–Machine Studies 19 (1983) 195–227.

[40] R.R. Yager, General multiple-objective decision functions and linguistically quantified statements, International Journal of Man–Machine Studies 21 (1984) 389–400.
[41] R.R. Yager, Interpreting linguistically quantified propositions, International Journal of Intelligent Systems 9 (1994) 541–569.
[42] M.S. Ying, The first-order fuzzy logic (I), in: Proc. 16th IEEE Int. Symposium on Multiple-Valued Logic, Virginia, 1986, pp. 242–247.
[43] M.S. Ying, On Zadeh's approach for interpreting linguistically quantified propositions, in: Proc. 18th IEEE Int. Symposium on Multiple-Valued Logic, Palma de Marlloca, 1988, pp. 248–254.
[44] M.S. Ying, Deduction theorem for many-valued inference, Zeitschrif fur Mathematische Logik und Grundlagen der Mathematik 37 (1991).
[45] M.S. Ying, The fundamental theorem of ultraproduct in Pavelka logic, Zeitschrif fur Mathematische Logik und Grundlagen der Mathematik 38 (1992) 197–201.
[46] M.S. Ying, Compactness, the Lowenheim–Sklom property and the direct-product of lattices of truth values, Zeitschrif fur Mathematische Logik und Grundlagen der Mathematik 38 (1992) 521–524.
[47] M.S. Ying, A logic for approximate reasoning, Journal of Symbolic Logic 59 (1994) 830–837.
[48] M.S. Ying, Perturbation of fuzzy reasoning, IEEE Transactions on Fuzzy Systems 7 (1999) 625–629.
[49] M.S. Ying, Implication operators in fuzzy logic, IEEE Transactions on Fuzzy Systems 10 (2002) 88–91.
[50] M.S. Ying, A formal model of computing with words, IEEE Transactions on Fuzzy Systems 10 (2002) 640–652.
[51] M.S. Ying, B. Bouchon-Meunier, Quantifiers, modifiers and qualifiers in fuzzy logic, Journal of Applied Non-Classical Logics 7 (1997) 335–342.
[52] M.S. Ying, B. Bouchon-Meunier, Approximate reasoning with linguistic modifiers, Internatonal Journal of Intelligent Systems 13 (1998).
[53] L.A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.
[54] L.A. Zadeh, Probability measures of fuzzy events, Journal of Mathematical Analysis and Applications 23 (1968) 421–427.
[55] L.A. Zadeh, PRUF—a meaning representation language for natural language, Internatonal Journal of Man–Machine Studies 10 (1978) 395–460.
[56] L.A. Zadeh, A computational approach to fuzzy quantifiers in natural language, Computers and Mathematics with Applications 9 (1983) 149–184.
[57] L.A. Zadeh, Fuzzy logic = computing with words, IEEE Transactions on Fuzzy Systems 4 (1996) 103–111.
[58] R. Zwick, E. Carlstein, D.V. Budescu, Measures of similarity among fuzzy concepts: A comparative analysis, International Journal of Approximate Reasoning 1 (1987) 221–242.

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Artificial Intelligence 170 (2006) 607–619
Artificial Intelligence
www.elsevier.com/locate/artint

Complexity of constructing solutions in the core based on synergies among coalitions ☆

Vincent Conitzer *, Tuomas Sandholm

Carnegie Mellon University, Computer Science Department, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

Received 5 December 2004; received in revised form 10 January 2006; accepted 16 January 2006

Available online 17 February 2006

Abstract

Coalition formation is a key problem in automated negotiation among self-interested agents, and other multiagent applications. A coalition of agents can sometimes accomplish things that the individual agents cannot, or can accomplish them more efficiently. Motivating the agents to abide by a solution requires careful analysis: only some of the solutions are stable in the sense that no group of agents is motivated to break off and form a new coalition. This constraint has been studied extensively in cooperative game theory: the set of solutions that satisfy it is known as the *core*. The computational questions around the core have received less attention. When it comes to coalition formation among software agents (that represent real-world parties), these questions become increasingly explicit.

In this paper we define a concise, natural, general representation for games in characteristic form that relies on superadditivity. In our representation, individual agents' values are given as well as values for those coalitions that introduce synergies. We show that this representation allows for efficient checking of whether a given outcome is in the core. We then show that determining whether the core is nonempty is NP-complete both with and without transferable utility. We demonstrate that what makes the problem hard in both cases is determining the collaborative possibilities (the set of outcomes possible for the grand coalition); we do so by showing that if these are given, the problem becomes solvable in time polynomial in the size of the representation in both cases. However, we then demonstrate that for a hybrid version of the problem, where utility transfer is possible only within the grand coalition, the problem remains NP-complete even when the collaborative possibilities are given. Finally, we show that for *convex* characteristic functions, a solution in the core can be computed efficiently (in $O(nl^2)$ time, where n is the number of agents and l is the number of synergies), even when the collaborative possibilities are not given in advance.

1. Introduction

Coalition formation is a key problem in automated negotiation among self-interested agents, and other multiagent applications. A coalition of agents can sometimes accomplish things that the individual agents cannot, or can accomplish them more efficiently. Motivating the agents to abide by a solution requires careful analysis: only some of the

☆ The material in this paper is based upon work supported by the National Science Foundation under CAREER Award IRI-9703122, Grant IIS-9800994, ITR IIS-0081246, ITR IIS-0121678, a Sloan Fellowship, and an IBM Ph.D. Fellowship. A short, early version of this paper appeared in the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03).

* Corresponding author.
E-mail addresses: conitzer@cs.cmu.edu (V. Conitzer), sandholm@cs.cmu.edu (T. Sandholm).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2006.01.005

solutions are stable in the sense that no group of agents is motivated to break off and form a new coalition. This constraint has been studied extensively in cooperative game theory: the set of solutions that satisfy it is known as the *core*. The computational questions around the core have received less attention. When it comes to coalition formation among software agents (that represent real-world parties), these questions become increasingly explicit.

While the driving motivation of this work is coalition formation among software agents, the computation of stable solutions may have applications beyond automated negotiation as well, for instance in electronic commerce. As an example, consider a large number of companies, some subsets of which could form profitable virtual organizations that can respond to larger or more diverse orders than the individual companies can. Determining stable value divisions allows one to see which potential virtual organizations would be viable in the sense that the companies in the virtual organization would naturally stay together. As another example, consider a future online service that determines how much each employee of a company should be paid so that the company does not collapse as a result of employees being bought away by other companies. The input to this service would be how much subsets of the company's employees would be paid if they left collectively (for instance, a whole department could be bought away). This input could come from salary databases or a manager's estimate. The computational problem of determining a stable renumeration would be crucial for such a service. Both of these example problems fit exactly under the model that we study in this paper.

The rest of the paper is organized as follows. In Section 2, we review the required concepts from cooperative game theory. In Section 3, we define a concise general representation for games in characteristic form that relies on superadditivity, and show that it allows for efficient (relative to the size of the representation) checking of whether a given outcome is in the core. In Section 4, we show that determining whether the core is nonempty is NP-complete both with and without transferable utility. In Section 5, we demonstrate that what makes the problem hard in both cases is determining the collaborative possibilities (the set of outcomes possible for the grand coalition); we do so by showing that if these are given, the problem becomes solvable in time polynomial in the size of the representation in both cases. In Section 6, we show that for a hybrid version of the problem, where utility transfer is possible only within the grand coalition, the problem remains NP-complete even when the collaborative possibilities are given. Finally, in Section 7, we show that if the characteristic function is convex, a solution in the core can be constructed efficiently (in $O(nl^2)$ time, where n is the number of agents and l is the number of synergies), even when the collaborative possibilities are not given in advance.

2. Definitions from cooperative game theory

In this section we review standard definitions from cooperative game theory, which we will use throughout the paper. In the definitions, we follow Mas-Colell et al. [21].

In general, how well agents in a coalition do may depend on what nonmembers of the coalition do (for example, see [2,6,10,22–24,28]). However, in cooperative game theory, coalition formation is usually studied in the context of *characteristic function games* where the utilities of the coalition members do not depend on the nonmembers' actions [5,16,31,35–37]. (One way to interpret this is to consider the coalition members' utilities to be the utilities they can *guarantee* themselves no matter what the nonmembers do [1].)

Definition 1. Given a set of agents A, a *utility possibility vector* u^B for $B = \{b_1, \ldots, b_{n_B}\} \subseteq A$ is a vector $(u_{b_1}, \ldots, u_{b_{n_B}})$ representing utilities that the agents in B can guarantee themselves by cooperating with each other. A *utility possibility set* is a set of utility possibility vectors for a given set B.

Definition 2. A *game in characteristic form* consists of a set of agents A and a utility possibility set $V(B)$ for each $B \subseteq A$.

Sometimes games in characteristic form have *transferable utility*, which means agents in a coalition can transfer utility among themselves. (Usually, this is done using monetary transfers.)

Definition 3. A game in characteristic form is said to have *transferable utility* if for every $B \subseteq A$ there is a number $v(B)$ (the *value* of B) such that $V(B) = \{u^B = (u^B_{b_1}, \ldots, u^B_{b_{n_B}})\colon \sum_{b \in B} u^B_b \leqslant v(B)\}$.

It is commonly assumed that the joining of two coalitions does not prevent them from acting as well as they could have acted separately. In other words, the composite coalition can coordinate by choosing not to coordinate. This assumption is known as *superadditivity*. When superadditivity holds, it is always best for the grand coalition of all agents to form.[1] We will assume superadditivity throughout the paper. This makes our hardness results *stronger* because even a restricted version of the problem is hard.

Definition 4. A game in characteristic form is said to be *superadditive* if, for any two disjoint subsets $B, C \subseteq A$, and for any $u^B \in V(B)$ and $u^C \in V(C)$, we have $(u^B, u^C) \in V(B \cup C)$. (In the case of transferable utility, this is equivalent to saying that for any $B, C \subseteq A$ with B and C disjoint, $v(B \cup C) \geqslant v(B) + v(C)$.)

We now need a solution concept. In this paper, we study only what is arguably the best known solution concept, namely the *core* (see, for example, [16,21,35]). It was first introduced by Gillies [13]. An outcome is said to be in the core if there is no subset of agents that can break off and form their own coalition in such a way that all of the agents in that coalition are better off. The following definition makes this precise.

Definition 5. An *outcome* $u^A = (u_1^A, \ldots, u_n^A) \in V(A)$ is *blocked* by coalition $B \subseteq A$ if there exists $u^B = (u_{b_1}^B, \ldots, u_{b_{n_B}}^B) \in V(B)$ such that for all $b \in B$, $u_b^B > u_b^A$. (In the case of transferable utility, this is equivalent to saying that the outcome is blocked by B if $v(B) > \sum_{b \in B} u_b^A$.) An outcome is in the *core* if it is blocked by no coalition.

In general, the core can be empty. If the core is empty, the game is inherently unstable because no matter what outcome is chosen, some subset of agents is motivated to pull out and form their own coalition. In other words, requiring that no subset of agents is motivated to break off into a coalition of its own overconstrains the system. An example of a game with an empty core is the one with agents $\{x, y, z\}$, where we have the utility possibility vectors $u^{\{x,y\}} = (2, 1)$, $u^{\{y,z\}} = (2, 1)$, and $u^{\{x,z\}} = (1, 2)$ (and the ones that can be derived from this through superadditivity). The same example with transferable utility also has an empty core.

In the rest of this paper, we will study the question of how complex it is to determine whether the core is nonempty, that is, whether there is a solution or the problem is overconstrained. First, as a basis for comparison, we mention the following straightforward, general algorithms for checking whether there exists a solution in the core. For a game without transferable utility, for every element $u^A \in V(A)$, check, for every coalition $B \subseteq A$, whether there exists an element $u^B \in V(B)$ such that u^B blocks u^A. For a game with transferable utility, we can use linear programming: we must decide on each agent's utility u_i in such a way that the total utility does not exceed $v(A)$, and such that for every coalition $B \subseteq A$, $v(B)$ does not exceed the total utility of the agents in B [25]. There are at least two reasons why these algorithms are not always practical. First, both algorithms are exponential in the number of agents, because they require a check or constraint for every subset of the agents.[2] Second, the first algorithm requires the computation of every utility possibility vector, and the second algorithm requires the computation of every value of a coalition. The computation of a single utility possibility vector or coalition value is not necessarily trivial, because it may require that the agents solve a complex collaborative planning problem.

Both of these two problems are intimately related to the *representation* of the characteristic form game. We cannot hope to compute solutions in the core in time that is less than exponential in the number of agents unless the representation is concise (that is, it does not simply list all the utility possibility vectors or all the coalition values). Moreover, we cannot address the question of how hard it is to compute individual utility possibility vectors or coalition values

[1] On the other hand, without superadditivity, even finding the optimal coalition structure (partition of agents into coalitions) can be hard [18,19, 29,32,33]. A third potential source of computational complexity (in addition to payoff/utility division as studied in this paper, and coalition structure generation) stems from each potential coalition having to solve a hard optimization problem in order to determine the value of the coalition. For example, when the agents are carrier companies with their own trucks and delivery tasks, they can save costs by forming a coalition (pooling their trucks and tasks), but each potential coalition faces a hard optimization problem: a vehicle routing problem defined by the coalition's trucks and tasks. The effect of such hard optimization problems on coalition formation has been studied by Sandholm and Lesser [30] and Tohmé and Sandholm [34].

[2] In the case of the first algorithm, there may also be exponentially or even infinitely many utility possibility vectors per coalition, making the algorithm even less satisfactory.

unless we know the representation from which they must be derived.[3] Because of this, in the next section, we introduce a concise representation of characteristic form games that is based on the synergies among coalitions.

3. Representing characteristic form games concisely

In our representation of games in characteristic form, we distinguish between games without transferable utility, where we specify some utility possibility vectors for some coalitions, and games with transferable utility, where we specify the values of some coalitions.

If the representation of the game specifies $V(B)$ or $v(B)$ explicitly for each coalition $B \subseteq A$, then the length of the representation is exponential in the number of agents. In that case, any algorithm for evaluating nonemptiness of the core (as long as it reads all the input) requires time exponential in the number of agents. However, that runtime is polynomial in the size of the input (this can be accomplished, for example, using the algorithms that we introduce in Section 5).

Most characteristic form games that represent real-world settings have some special structure. This usually allows for a game representation that is significantly more concise. The complexity of characterizing the core has already been studied in certain specific concisely expressible families of games. For example, Deng and Papadimitriou [9] study a game in which the agents are vertices of a graph with weights on the edges, and the value of a coalition is determined by the total weight of the edges contained in the subgraph induced by that coalition. Faigle et al. [12] study a game in which the agents are vertices on a graph, and the cost of a coalition is the weight of a *minimum spanning tree* for the subgraph induced by that coalition. They show that determining whether a solution is in the core is NP-complete in this game. In other work [11], the same authors study a modification of this game where the cost of a coalition is the minimum length of a traveling salesman tour for the subgraph induced by that coalition, and show how to use linear programming to give a solution that is *approximately* in the core (in the sense that every coalition pays at most $4/3$ times their own cost). Markakis and Saberi [20] study how to construct a solution in the core for a *multicommodity flow game* on graphs (defined by Papadimitriou [27]). In this game, the agents correspond to the vertices of the graph, and each agent's utility is the total flow originated or terminated at that agent's vertex. A coalition can obtain any utility vector corresponding to a feasible flow for the subgraph induced by that coalition. Markakis and Saberi show that the core is always nonempty in such games, and they show that for the case of transferable utility, a solution in the core can be constructed in polynomial time in such games. Goemans and Skutella [14] study how to construct a solution in the core for a *facility location game*. In this game, several facilities must be opened (at a cost), and the agents must be connected to them (at a cost). Here, a solution is in the core if no subset of agents has an incentive to deviate and build the facilities and connections on their own. Goemans and Skutella show that there is a solution in the core if and only if there is no integrality gap for a particular linear programming relaxation (which is NP-complete to verify). Pal and Tardos [26] and Gupta et al. [15] study a game in which the agents need to be connected to a common source in a graph, and give a polynomial-time algorithm for constructing a solution that is approximately in the core. Finally, Deng et al. [8] study an integer programming formulation which captures many games on graphs, and (as in the result by Goemans and Skutella [14]) show that the core is nonempty if and only if the linear programming relaxation has an integer optimal solution. All of those results depend on concise game representations which are specific to the game families under study. Typically, such a family of games is played on a combinatorial structure such as a graph. Cooperative games on combinatorial structures have been systematically studied [3].

As a point of deviation, we study a natural representation that can capture *any* characteristic form game.[4] Conciseness in our representation stems only from the fact that in many settings, the synergies among coalitions are sparse. When a coalition introduces no new synergy, its utility possibility vectors can be *derived* using superadditivity. Therefore, the input needs to include only the utility possibility vectors of coalitions that introduce synergy. The definitions below make this precise. We first present our representation of games without transferable utility.

[3] It should be remarked that deriving utility possibility vectors or coalition values from a given representation may be only part of the problem: it is possible that this representation itself must be constructed from another, even more basic representation, and this construction may itself be computationally nontrivial.

[4] Our hardness results are not implied by the earlier hardness results for specific game families, because the games must be concisely representable in our input language to prove our hardness results.

Definition 6. We represent a game in characteristic form without transferable utility by a set of agents A, and a set of utility possibility vectors $W = \{(B, u^{B,k})\}$. (Here there may be multiple vectors for the same B, distinguished by different k indices.) The utility possibility set for a given $B \subseteq A$ is then given by $V(B) = \{u^B\colon\ u^B = (u^{B_1}, \ldots, u^{B_r}), \bigcup_{1 \leqslant i \leqslant r} B_i = B$, all the B_i are disjoint, and for all the $B_i, (B_i, u^{B_i}) \in W\}$. To avoid senseless cases that have no outcomes, we also require that $(\{a\}, (0)) \in W$ for all $a \in A$.[5]

Listing all the utility possibility vectors is of course not always feasible, and when it is not feasible, another representation should be used. For example, in a game with transferable utility, there are infinitely many (actually, a continuum of) utility possibility vectors. Therefore, we give a more appropriate representation of games with transferable utility below. Nevertheless, there do exist settings in which all the utility possibility vectors can be listed explicitly. For example, we may have historical data on which coalitions have formed in the past and what the agents' utilities were in those coalitions. At the minimum, we should ensure that no subcoalition will collectively yearn for times past (and perhaps seek to recreate them by breaking off from the present-day coalition). In this case, we should try to find a solution that is in the core of the game given by the historical characteristic function.

We now present our representation of games with transferable utility.

Definition 7. We represent a game in characteristic form with transferable utility by a set of agents A, and a set of values $W = \{(B, v(B))\}$. The value for a given $B \subseteq A$ is then given by $v(B) = \max\{\sum_{1 \leqslant i \leqslant r} v(B_i)\colon \bigcup_{1 \leqslant i \leqslant r} B_i = B$, all the B_i are disjoint, and for all the $B_i, (B_i, v(B_i)) \in W\}$. To avoid senseless cases that have no outcomes, we also require that $(\{a\}, 0) \in W$ whenever $\{a\}$ does not receive a value elsewhere in W.

Thus, we only need to specify a basis of utility possibilities, from which we can then derive the others. (This derivation can, however, be computationally hard, as we will discuss shortly.) This representation integrates rather nicely with real-world problems where determining any coalition's value is complex. For example, in the multiagent vehicle routing problem, we solve the routing problem for every coalition that might introduce new synergies. When it is clear that there is no synergy between two coalitions (for example, if they operate in different cities and each one only has deliveries within its city), there is no need to solve the routing problem of the coalition that would result if the two coalitions were to merge.

The following lemmas indicate that we can also use this representation effectively for checking whether an outcome is in the core, that is, whether it satisfies the strategic constraints.

Lemma 1. *Without transferable utility, an outcome $u^A = (u_1^A, \ldots, u_n^A) \in V(A)$ is blocked by some coalition if and only if it is blocked by some coalition B through some utility vector u^B, where $(B, u^B) \in W$.*

Proof. The "if" part is trivial. For "only if", suppose u^A is blocked by coalition C through some u^C, so that for every $c \in C$, $u_c^C > u_c^A$. We know $u^C = (u^{C_1}, \ldots, u^{C_r})$ where $(C_i, u^{C_i}) \in W$. But then, C_1 blocks u^A through u^{C_1}. □

The proof for the same lemma in the case of transferable utility is only slightly more intricate.

Lemma 2. *With transferable utility, an outcome $u^A = (u_1^A, \ldots, u_n^A) \in V(A)$ is blocked by some coalition if and only if it is blocked by some coalition B through its value $v(B)$, where $(B, v(B)) \in W$.*

Proof. The "if" part is trivial. For "only if", suppose u^A is blocked by coalition C through $v(C)$, so that $v(C) > \sum_{c \in C} u_c^A$. We know that $v(C) = \sum_{1 \leqslant i \leqslant r} v(C_i)$ where $(C_i, v(C_i)) \in W$. It follows that $\sum_{1 \leqslant i \leqslant r} v(C_i) > \sum_{1 \leqslant i \leqslant r} \sum_{c \in C_i} u_c^A$, and hence for at least one C_i, we have $v(C_i) > \sum_{c \in C_i} u_c^A$. But then C_i blocks u^A through $v(C_i)$. □

One drawback is that under this representation, it is, in general, NP-complete to compute a given coalition's value. Thus, the representation still leaves part of the collaborative optimization problem of determining a coalition's value to

[5] Setting the utility to 0 in this case is without loss of generality, as we can simply normalize the utility function to obtain this.

be solved (which is not entirely unnatural given that in many practical settings, the collaborative optimization problem is very complex). Rather than prove this NP-hardness here directly, we note that it follows from results in the coming sections, as we will show.

4. Checking whether the core is nonempty is hard

We now show that with this representation, it is hard to check whether the core is nonempty. This holds both for the nontransferable utility setting and for the transferable utility setting.

Definition 8 *(CORE-NONEMPTY).* We are given a superadditive game in characteristic form (with or without transferable utility) in our representation language. We are asked whether the core is nonempty.

We will demonstrate NP-hardness of this problem by reducing from the NP-complete EXACT-COVER-BY-3-SETS problem [17].

Definition 9 *(EXACT-COVER-BY-3-SETS).* We are given a set S of size $3m$ and a collection of subsets $\{S_i\}_{1 \leqslant i \leqslant r}$ of S, each of size 3. We are asked if there is a cover of S consisting of m of the subsets.

The intuition behind both NP-hardness reductions from EXACT-COVER-BY-3-SETS presented in this section is the following. We let the elements of S correspond to some of the agents, and we let the subsets S_i correspond to some of the elements of W, in such a way that the grand coalition can achieve a "good" outcome if and only if we can cover all these agents with a subset of disjoint elements of W—that is, if and only if an exact cover exists. Then, we add some additional agents and elements of W such that only such a good outcome would be strategically stable (this is the nontrivial part of both proofs, which is why these results do not follow trivially from the hardness of solving a coalition's collaborative optimization problem). The formal proofs follow.

Theorem 1. *CORE-NONEMPTY without transferable utility is NP-complete.*

Proof. To show that the problem is in NP, nondeterministically choose a subset of W, and check if the corresponding coalitions constitute a partition of A. If so, check if the outcome corresponding to this partition is blocked by any element of W.

To show NP-hardness, we reduce an arbitrary EXACT-COVER-BY-3-SETS instance to the following CORE-NONEMPTY instance. Let the set of agents be $A = S \cup \{w, x, y, z\}$. For each S_i, let (S_i, u^{S_i}) be an element of W, with $u^{S_i} = (2, 2, 2)$. Also, for each $s \in S$, let $(\{s, w\}, u^{\{s,w\}})$ be an element of W, with $u^{\{s,w\}} = (1, 4)$. Also, let $(\{w, x, y, z\}, u^{\{w,x,y,z\}})$ be an element of W, with $u^{\{w,x,y,z\}} = (3, 3, 3, 3)$. Finally, let $(\{x, y\}, u^{\{x,y\}})$ with $u^{\{x,y\}} = (2, 1)$, $(\{y, z\}, u^{\{y,z\}})$ with $u^{\{y,z\}} = (2, 1)$, $(\{x, z\}, u^{\{x,z\}})$ with $u^{\{x,z\}} = (1, 2)$ be elements of W. The only other elements of W are the required ones giving utility 0 to singleton coalitions. We claim the two instances are equivalent.

First suppose there is an exact cover by 3-sets consisting of $S_{c_1}, \ldots, S_{c_m}$. Then the following outcome is possible: $(u^{S_{c_1}}, \ldots, u^{S_{c_m}}, u^{\{w,x,y,z\}}) = (2, 2, \ldots, 2, 3, 3, 3, 3)$. It is easy to verify that this outcome is not blocked by any coalition. It follows that the core is nonempty.

Now suppose there is no exact cover by 3-sets. Suppose the core is nonempty, that is, it contains some outcome $u^A = (u^{C_1}, \ldots, u^{C_r})$ with each (C_i, u^{C_i}) an element of W, and the C_i disjoint. Then one of the C_i must be $\{s, w\}$ for some $s \in S$: for if this were not the case, there must be some $s \in S$ with $u^A_s = 0$, because the C_i that are equal to S_i cannot cover S; but then $\{s, w\}$ would block the outcome. Thus, none of the C_i can be equal to $\{w, x, y, z\}$. Then one of the C_i must be one of $\{x, y\}, \{y, z\}, \{x, z\}$, or else two of $\{x, y, z\}$ would block the outcome. By symmetry, we can without loss of generality assume it is $\{x, y\}$. But then $\{y, z\}$ will block the outcome. (Contradiction.) It follows that the core is empty. □

We might hope that the convexity introduced by transferable utility makes the problem tractable through, for example, linear programming. This turns out not to be the case.

Theorem 2. *CORE-NONEMPTY with transferable utility is NP-complete.*

Proof. To show that the problem is in NP, nondeterministically choose a subset of W, and check if the corresponding coalitions constitute a partition of A. If so, nondeterministically divide the sum of the coalitions' values over the agents, and check if this outcome is blocked by any element of W.

To show NP-hardness, we reduce an arbitrary EXACT-COVER-BY-3-SETS instance to the following CORE-NONEMPTY instance. Let the set of agents be $A = S \cup \{x, y\}$. For each S_i, let $(S_i, 3)$ be an element of W. Additionally, let $(S \cup \{x\}, 6m)$, $(S \cup \{y\}, 6m)$, and $(\{x, y\}, 6m)$ be elements of W. The only other elements of W are the required ones giving value 0 to singleton coalitions. We claim the two instances are equivalent.

First suppose there is an exact cover by 3-sets consisting of $S_{c_1}, \ldots, S_{c_m}$. Then the value of coalition S is at least $\sum_{1 \leqslant i \leqslant m} v(S_{c_i}) = 3m$. Combining this with the coalition $\{x, y\}$, which has value $6m$, we conclude that the grand coalition A has value at least $9m$. Hence, the outcome $(1, 1, \ldots, 1, 3m, 3m)$ is possible. It is easy to verify that this outcome is not blocked by any coalition. It follows that the core is nonempty.

Now suppose there is no exact cover by 3-sets. Then the coalition S has value less than $3m$ (since there are no m disjoint S_i), and as a result the value of the grand coalition is less than $9m$. It follows that in any outcome, the total utility of at least one of $S \cup \{x\}$, $S \cup \{y\}$, and $\{x, y\}$ is less than $6m$. Hence, this coalition will block. It follows that the core is empty. □

Our results imply that in the general case, it is computationally hard to make any strategic assessment of a game in characteristic form when it is concisely represented.

5. Specifying information about the grand coalition makes the problem tractable

Our proofs that CORE-NONEMPTY is hard relied on constructing instances where it is difficult to determine what the grand coalition can accomplish. In effect, the hardness derived from the fact that even collaborative optimization is hard in these instances. While this is indeed a real difficulty that occurs in the analysis of characteristic form games, we may nevertheless wonder to what extent computational complexity issues are introduced by the purely strategic aspect of the games. To analyze this, we investigate the computational complexity of CORE-NONEMPTY when $V(A)$ (or $v(A)$) is *explicitly* provided as input, so that determining what the grand coalition can accomplish can no longer be the source of any complexity.[6] It indeed turns out that the problem becomes easy both with and without transferable utility.

Theorem 3. *When $V(A)$ is explicitly provided, CORE-NONEMPTY without transferable utility is in P.*

Proof. The following simple algorithm accomplishes this efficiently. For each element of $V(A)$, check whether it is blocked by any element of W. The algorithm is correct due to Lemma 1. □

For the transferable utility case, we make use of linear programming.

Theorem 4. *When $v(A)$ is explicitly provided, CORE-NONEMPTY with transferable utility is in P.*

Proof. We decide how to allocate the $v(A)$ among the agents by solving a linear program. Because of Lemma 2, the core is nonempty if and only if the following linear program has a solution:

$$\sum_{1 \leqslant i \leqslant n} u_i \leqslant v(A); \quad \forall \big(B, v(B)\big) \in W, \quad \sum_{b \in B} u_b \geqslant v(B). \tag{1}$$

Because the size of the program is polynomial, this constitutes a polynomial-time algorithm. □

[6] Bilbao et al. [4] have studied the complexity of the core in characteristic form games with transferable utility when there is an oracle that can provide the value $v(B)$ of any coalition B. Our amended input corresponds to asking one such query in addition to obtaining the unamended input.

The algorithms in the proofs also construct a solution that is in the core, if the core is nonempty. We note that we have now also established that computing $v(A)$ is NP-hard, because, apart from the computation of $v(A)$, we have an efficient algorithm for computing a solution in the core—which we have already shown is NP-hard.

6. Games with limited utility transfer remain hard

Not all complexity issues disappear through having the collaborative optimization problem solution available. It turns out that if we allow for games *with limited utility transfer*, where only *some* coalitions can transfer utility among themselves, the hardness returns. In particular, we show hardness in the case where only the grand coalition can transfer utility. This is a natural model for example in settings where there is a market institution that enforces payments, but if a subset of the agents breaks off, the institution collapses and payments cannot be enforced. Another example application is the setting described in Section 3 in which we wish to ensure that there is no subcoalition such that all of its agents were happier in a past time period when they formed a coalition by themselves.

Definition 10. We represent a game in characteristic form in which only the grand coalition can transfer utility by a set of agents A, a set of utility possibility vectors $W = \{(B, u^{B,k})\}$, and a value $v(A)$ for the grand coalition. The utility possibility set for a given $B \subseteq A$, $B \neq A$ is then given by $V(B) = \{u^B\colon u^B = (u^{B_1}, \ldots, u^{B_r}), \bigcup_{1 \leqslant i \leqslant r} B_i = B,$ all the B_i are disjoint, and for all the B_i, $(B_i, u^{B_i}) \in W\}$. To avoid senseless cases that have no outcomes, we also require that $(\{a\}, (0)) \in W$ for all $a \in A$. For the grand coalition A, any vector of utilities that sum to at most $v(A)$ is possible.

We demonstrate that the CORE-NONEMPTY problem is NP-hard in this setting by reducing from the NP-complete VERTEX-COVER problem [17].

Definition 11 *(VERTEX-COVER).* We are given a graph $G = (V, E)$, and a number k. We are asked whether there is a subset of V of size k such that each edge has at least one of its endpoints in the subset.

We are now ready to state our result.

Theorem 5. *When only the grand coalition can transfer utility, CORE-NONEMPTY is NP-complete* (*even when* $v(A)$ *is explicitly provided as input*).

Proof. To show that the problem is in NP, nondeterministically divide $v(A)$ over the agents, and check if this outcome is blocked by any element of W.

To show NP-hardness, we reduce an arbitrary VERTEX-COVER instance to the following CORE-NONEMPTY instance. Let $A = V \cup \{x, y, z\}$, and let $v(A) = 6|V| + k$. Furthermore, for each edge (v_i, v_j), let $(\{v_i, v_j\}, u^{\{v_i,v_j\}})$ be an element of W, with $u^{\{v_i,v_j\}} = (1, 1)$. Finally, for any $a, b \in \{x, y, z\}$ $(a \neq b)$, let $(\{a, b\}, u^{\{a,b\}})$ be an element of W, with $u^{\{a,b\}} = (3|V|, 2|V|)$. The only other elements of W are the required ones giving utility 0 to singleton coalitions. This game does not violate the superadditivity assumption, since without the explicit specification of $v(A)$, superadditivity can at most imply that $v(A) = 6|V| \leqslant 6|V| + k$. We claim the two instances are equivalent.

First suppose there is a vertex cover of size k. Consider the following outcome: all the vertices in the vertex cover receive utility 1, all the other vertices receive utility 0, and each of x, y, and z receives utility $2|V|$. Using the fact that all the edges are covered, it is easy to verify that this outcome is not blocked by any coalition. It follows that the core is nonempty.

Now suppose there is some outcome u^A in the core. In such an outcome, either each of $\{x, y, z\}$ receives at least $2|V|$, or two of them receive at least $3|V|$ each. (For if not, there is some $a \in \{x, y, z\}$ with $u^A_a < 2|V|$ and some $b \in \{x, y, z\}$ $(b \neq a)$ with $u^A_b < 3|V|$, and the coalition $\{b, a\}$ will block through $u^{\{b,a\}} = (3|V|, 2|V|)$.) It follows that the combined utility of all the elements of V is at most k. Now, for each edge (v_i, v_j), at least one of its vertices must receive utility at least 1, or this edge would block. Thus, the vertices that receive at least 1 cover the edges. But because the combined utility of all the elements of V is at most k, there can be at most k such vertices. It follows that there is a vertex cover. □

Games in which only some coalitions can transfer utility are quite likely to appear in real-world multiagent settings, for example because only some of the agents use a currency. Our result shows that for such games, even when the collaborative optimization problem has already been solved, it can be computationally hard to strategically assess the game.

7. A fast algorithm for convex characteristic functions

While we have shown that constructing a solution in the core is hard in general, we may wonder whether restrictions on the characteristic function make the problem easier. In this section, we show that if the characteristic function is *convex* (in which case it is known that the core is always nonempty), a solution in the core can be constructed efficiently (in $O(nl^2)$ time, where n is the number of agents and l is the number of synergies) under our representation. We do not require that the value of the grand coalition is known, nor do we need to use techniques from linear programming.

We will restrict our attention to games with transferable utility in this section. A characteristic function v is *convex* if for any coalitions B and C, we have $v(B \cup C) - v(C) \geqslant v(B) - v(B \cap C)$ [25]. That is, the marginal contribution of a subcoalition to a larger set is always larger (or at least as large). One natural setting in which the characteristic function is convex is the following. Suppose that every agent i has a set of *skills* S_i, and suppose that every agent's skills are unique, that is, $S_i \cap S_j = \{\}$ for all $i \neq j$. Furthermore, suppose that there is a set of *tasks* T that the agents can accomplish, and that each task $t \in T$ has a value $v(t) \geqslant 0$, as well as a set of skills $S(t)$ that are required to accomplish t. Then, the value of a coalition B is the sum of the values of the tasks that they can accomplish, that is, $v(B) = \sum_{t \in T:\ S(t) \subseteq \bigcup_{i \in B} S_i} v(t)$. In this setting, the marginal set of tasks that can be accomplished by adding a given subset of the agents to an existing coalition of agents is increasing in the existing coalition (as the existing coalition gets larger, the marginal set of tasks that can be accomplished by adding the given subset of agents can get no smaller), and therefore the characteristic function is convex. One example application is the following: suppose each agent holds some medical patents (these are the agents' "skills"), and there is a set of drugs (the "tasks"), each of which has a market value and requires a subset of the patents to be manufactured.

When the characteristic function is convex, one method of constructing a solution in the core is the following: impose an arbitrary order on the agents, add them to the coalition one at a time, and give each agent its marginal value (see, for example, Osborne and Rubinstein [25]). Unfortunately, this scheme cannot be applied directly under our representation, as it requires knowing the values of the relevant coalitions. As we have seen earlier, computing these values is hard for general characteristic functions. However, in this section we will show that when convexity holds, this is no longer the case, and solutions in the core can be constructed efficiently.

As it turns out, it is easier to think about the implications of convexity for our representation if we first assume that there are no explicitly specified pairs $(B, v(B))$ that are redundant. That is, if $v(B)$ could have been inferred from the values of other coalitions by superadditivity, we assume that it is not explicitly specified. Thus, only the *synergetic* coalitions' values are explicitly specified, where a coalition is synergetic if its value could not have been derived from the values of its proper subcoalitions. We will later show how to remove this assumption. Formally:

Definition 12. A coalition B is *synergetic* if $v(B) > \max\{\sum_{1 \leqslant i \leqslant r} v(B_i)$: $\bigcup_{1 \leqslant i \leqslant r} B_i = B$, each B_i is a proper subset of B, all the B_i are disjoint, and for all the B_i, $(B_i, v(B_i)) \in W\}$.

First we show that the union of two intersecting synergetic coalitions must be synergetic when convexity holds.

Lemma 3. *Let v be a convex characteristic function. Suppose B and C are both synergetic coalitions (for v), with a nonempty intersection ($B \cap C \neq \{\}$). Then $B \cup C$ is a synergetic coalition.*

Proof. If one of B and C is contained in the other, the lemma is trivially true. Hence, we can assume that this is not the case, that is, $B - C \neq \{\}$ and $C - B \neq \{\}$.

First suppose that $v(B \cup C) = v(B) + v(C - B)$. Then, $v(B \cup C) - v(C) = v(B) + v(C - B) - v(C) < v(B) + v(C - B) - v(C - B) - v(B \cap C) = v(B) - v(B \cap C)$ (where the inequality follows from C being a synergetic coalition), contradicting convexity. Hence, $v(B \cup C) > v(B) + v(C - B)$. By symmetry, $v(B \cup C) > v(C) + v(B - C)$.

We will now prove the lemma by induction on $|B \cup C|$. For $|B \cup C| \leqslant 2$ the lemma is vacuously true. Now suppose that it is always true for $|B \cup C| < k$, but it fails for $|B \cup C| = k$. That is, we have some B and C (both synergetic,

with a nonempty intersection and neither contained in the other), where $B \cup C$ is not synergetic. Without loss of generality, we assume that B is maximal, that is, there is no strict superset of B in $B \cup C$ that is also synergetic. Let $D_1, D_2, \ldots, D_n$ be a partition of $B \cup C$ (with $n > 1$) such that each D_i is synergetic and $\sum_{i=1}^{n} v(D_i) = v(B \cup C)$ (such a partition must exist because we are supposing $B \cup C$ is not synergetic). If one of the D_i were equal to either B or C, then we would have $v(B \cup C) = v(B) + v(C - B)$ or $v(B \cup C) = v(C) + v(B - C)$, which we showed impossible at the beginning of the proof. Moreover, it is not possible that all the D_i are either contained in B, or have an empty intersection with B. For if this were the case, then $v(B \cup C) = \sum_{i=1}^{n} v(D_i) = \sum_{D_i \subseteq B} v(D_i) + \sum_{D_i \subseteq C-B} v(D_i) < v(B) + \sum_{D_i \subseteq C-B} v(D_i) \leqslant v(B \cup C)$ (where the first inequality derives from the facts that B is synergetic and no D_i is equal to B). Thus, there is some D_i^* with $D_i^* \cap B \neq \{\}$ and $D_i^* \cap (C - B) \neq \{\}$. Suppose that this D_i^* does not contain all of $C - B$. Then, we can apply the induction assumption to B and D_i^* to conclude that $B \cup D_i^*$ is synergetic. But this contradicts the maximality of B. On the other hand, suppose that D_i^* does contain all of $C - B$. Then $B \cup D_i^* = B \cup C$, and $v(B \cup D_i^*) = v(D_i^*) + v(B - D_i^*)$, which we showed impossible at the beginning of the proof. □

Next, we show that any partition of B into maximal synergetic coalitions will give us the value of B when convexity holds.

Lemma 4. *If v is convex, for any coalition B, for any partition $D_1, D_2, \ldots, D_n$ of B where each D_i is synergetic and maximal in B (there is no superset of D_i in B which is also synergetic), $v(B) = \sum_{i=1}^{n} v(D_i)$.*

Proof. Suppose not. Let $E_1, E_2, \ldots, E_m$ be a partition of B where each E_j is synergetic and $\sum_{j=1}^{m} v(E_j) = v(B)$. No E_j can be a strict superset of a D_i, because the D_i are maximal. Also, it cannot be the case that every E_j is a subset of some D_i, because then we would have $\sum_{j=1}^{m} v(E_j) < \sum_{i=1}^{n} v(D_i) < v(B) = \sum_{j=1}^{m} v(E_j)$. It follows that there is some E_j^* that is not contained in any D_i. Some D_i^* must intersect with E_j^*. But then by Lemma 3, $D_i^* \cup E_j^*$ must be synergetic, contradicting that D_i^* is maximal. □

We now show how to quickly compute the value of any coalition when convexity holds, presuming that there are no redundant specifications of coalition values. (That is, all the coalitions with explicitly specified values are synergetic. We will show how to remove this assumption shortly.)

Lemma 5. *Suppose v is convex. For any coalition B, if all its subcoalitions $D \subseteq B$ for which $v(D)$ is explicitly specified are synergetic (that is, there are no redundant specifications), the explicitly specified $(D, v(D))$ pairs are sorted by $|D|$, and each subset D is represented as a sorted list of agents, then $v(B)$ can be computed in $O(nl)$ time by computing a partition of maximal synergetic subcoalitions. (Here, n is the number of agents and l is the number of explicitly specified $(D, v(D))$ pairs.)*

Proof. Because all synergetic subcoalitions must be explicitly specified, we know that the set of subcoalitions D for which $(D, v(D))$ is explicitly specified is exactly the set of synergetic subcoalitions. Thus, by Lemma 4, we merely need to find a partition of B consisting of maximal subcoalitions whose value is explicitly specified. To do so, we will add such maximal subcoalitions to our partition one at a time. Because the explicitly specified $(D, v(D))$ pairs are sorted by $|D|$, we can read through them starting with those with the highest $|D|$ and work our way downwards to the ones with smallest $|D|$. When we arrive at any given D, we add it to the partition if and only if no members of D were previously added to the partition (as a member of another subcoalition). (We can perform each such check in $O(n)$ time, by keeping track of the set of agents that have already been added to the partition as a sorted list. To see if this set intersects with D, which is also stored as a sorted list, we go through the two sorted lists in parallel (as in, for instance, the well-known MergeSort algorithm), which requires $O(n)$ time. Updating the list of all agents that have already been added to a coalition when a given set D is added to the partition can be done in $O(n)$ time as well, again going through the two lists in parallel, and merging them.) Every synergetic D that we add to the partition must be maximal: for, if D is not maximal, when we arrive at D, the maximal synergetic D' containing it must already have been added, for the following reason. By Lemma 3 a maximal synergetic D' cannot intersect with another synergetic subcoalition of equal or larger size. Hence none of the members of D' were already in some element of the partition when we were considering adding D', and thus we must have added D'. Moreover, by the same reasoning, any member of B

must eventually be in some element of the partition (because the maximal synergetic subcoalition containing it must be added). Thus we will get a valid partition. □

Of course, for generality, we would like to allow for redundant specifications. Next, we show that we can indeed allow for them, because we can quickly remove any redundant specifications when convexity holds.

Lemma 6. *If v is convex and each subset $B \subseteq A$ is represented as a sorted list of agents, then the set of explicitly specified $(B, v(B))$ pairs can be reduced to the subset of only the synergetic ones in $O(nl^2)$ time. (Here, n is the number of agents and l is the number of explicitly specified $(B, v(B))$ pairs.)*

Proof. After sorting the $(B, v(B))$ pairs by $|B|$ (which takes $O(l \log(l))$ time), we will go through them starting at those with the smallest $|B|$ and working our way up to the larger ones, eliminating each nonsynergetic one, as follows. We observe that when we arrive at the pair $(B, v(B))$, all the remaining explicitly specified $(D, v(D))$ with $D \subseteq B$ will be synergetic (apart from possibly B itself). Thus, we can temporarily remove $(B, v(B))$, then according to Lemma 5 compute the new value of B (say, $v'(B)$) in $O(nl)$ time, and remove the pair $(B, v(B))$ if and only if $v'(B) = v(B)$. (We observe that, if B is indeed synergetic, removing the pair $(B, v(B))$ may break the convexity, in which case the algorithm from Lemma 5 may select the wrong maximal subcoalitions and compute too small a value for $v'(B)$. Of course, in this case the computed $v'(B)$ is still smaller than $v(B)$ and we will not remove the pair $(B, v(B))$, as required.) □

Finally, we show how to quickly compute a solution in the core when convexity holds. To distribute the value, we make use of the following known lemma mentioned at the beginning of this section.

Lemma 7. (Osborne and Rubinstein [25].) *For any order imposed on the agents, if we add the agents to the coalition one at a time, and give each agent its marginal value, then, if the characteristic function is convex, we obtain a solution in the core.*

This leads to the following result.

Theorem 6. *If v is convex, a solution in the core can be constructed in $O(nl^2)$ time. (Here, n is the number of agents and l is the number of explicitly specified $(B, v(B))$ pairs.)*

Proof. If the subsets $B \subseteq A$ are not yet represented as sorted lists, we can correct this in $O(nl \log(n))$ time. Then, we remove all the redundant $(B, v(B))$ pairs using Lemma 6, and sort the remaining ones by $|B|$, in $O(nl^2)$ time. Then (using the scheme from Lemma 7) we order the members of the grand coalition in an arbitrary way, add them in one by one, and give each of them their marginal value. This requires the computation of the value of one additional coalition for each agent (namely, the coalition that results from adding that agent), which takes $O(nl)$ time per agent using Lemma 5, for a total of $O(n^2 l)$ time. Because every agent appears in at least one synergetic subcoalition, and hence $n \leqslant l$, the whole algorithm runs in $O(nl^2)$ time. □

Another polynomial-time approach for constructing a solution in the core that does not rely on Lemma 7 is to compute the value of the grand coalition (using Lemmas 6 and 5), and then solve the linear programming problem as in Section 5.

8. Conclusions and future research

As we have argued, coalition formation is a key problem in automated negotiation among self-interested agents, and other multiagent applications. A coalition of agents can sometimes accomplish things that the individual agents cannot, or can accomplish them more efficiently. Motivating the agents to abide by a solution requires careful analysis: only some of the solutions are stable in the sense that no group of agents is motivated to break off and form a new coalition. This constraint has been studied extensively in cooperative game theory, but the computational questions

around this constraint have received less attention. When it comes to coalition formation among software agents (that represent real-world parties), these questions become increasingly explicit.

In this paper we defined a concise general representation for games in characteristic form that relies on superadditivity, and showed that it allows for efficient checking of whether a given outcome is in the core. We then showed that determining whether the core is nonempty is NP-complete both with and without transferable utility. We demonstrated that what makes the problem hard in both cases is determining the collaborative possibilities (the set of outcomes possible for the grand coalition); we did so by showing that if these are given, the problem becomes solvable in time polynomial in the size of the representation in both cases. However, we then demonstrated that for a hybrid version of the problem, where utility transfer is possible only within the grand coalition, the problem remains NP-complete even when the collaborative possibilities are given. Finally, we showed that if the characteristic function is convex, a solution in the core can be constructed efficiently (in $O(nl^2)$ time, where n is the number of agents and l is the number of synergies), even when the collaborative possibilities are not given in advance.

Future research can take a number of different directions. One such direction is to investigate restricted families of games in characteristic form (such as the family of convex games, which we analyzed in this paper). Another direction is to evaluate other solution concepts in cooperative game theory from the perspective of computational complexity under our input representation. A long-term goal is to extend the framework for finding a strategically stable solution to take into account issues of computational complexity in determining the synergies among coalitions [30,34]. For example, nontrivial routing problems may need to be solved (potentially only approximately) in order to determine the synergies. How to efficiently construct a stable solution when the cost of determining a synergy is significant (and possibly varying across coalitions) is an important open problem. Finally, we can investigate the same questions under different representations. These representations can be for restricted domains only (we discussed some results along this line in Section 3); they can also be general, so that the representation is concise only on some instances (ideally those that are likely to occur in practice). In the latter category, we have recently introduced and studied another representation that allows for the decomposition of the characteristic function over distinct *issues*, each of which concerns only a small number of the agents [7].

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments.

References

[1] R. Aumann, Acceptable points in general cooperative n-person games, in: Contributions to the Theory of Games, vol. IV, in: Annals of Mathematics Study, vol. 40, Princeton University Press, Princeton, NJ, 1959, pp. 287–324.
[2] B.D. Bernheim, B. Peleg, M.D. Whinston, Coalition-proof Nash equilibria: I. Concepts, J. Econ. Theory 42 (1) (1987) 1–12.
[3] J.M. Bilbao, Cooperative Games on Combinatorial Structures, Kluwer Academic, Dordrecht, 2000.
[4] J.M. Bilbao, J.R. Fernandez, J.J. Lopez, Complexity in cooperative game theory, 2000.
[5] A. Charnes, K.O. Kortanek, On balanced sets, cores, and linear programming, Technical Report 12, Cornell Univ., Dept. of Industrial Eng. and Operations Res., Ithaca, NY, 1966.
[6] K. Chatterjee, B. Dutta, D. Ray, K. Sengupta, A noncooperative theory of coalitional bargaining, Rev. Econ. Stud. 60 (1993) 463–477.
[7] V. Conitzer, T. Sandholm, Computing Shapley values, manipulating value division schemes, and checking core membership in multi-issue domains, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), San Jose, CA, 2004, pp. 219–225.
[8] X. Deng, T. Ibaraki, H. Nagamochi, Algorithms and complexity in combinatorial optimization games, in: Proceedings of the Eighth Annual ACM–SIAM Symposium on Discrete Algorithms, 1997, pp. 720–729.
[9] X. Deng, C.H. Papadimitriou, On the complexity of cooperative solution concepts, Math. Oper. Res. (1994) 257–266.
[10] R. Evans, Coalitional bargaining with competition to make offers, Games Econ. Behav. 19 (1997) 211–220.
[11] U. Faigle, S. Fekete, W. Hochstattler, W. Kern, Approximating the core of Euclidean TSP games, Oper. Res. (1993) 152–156.
[12] U. Faigle, S. Fekete, W. Hochstattler, W. Kern, On the complexity of testing membership in the core of min-cost spanning trees, Internat. J. Game Theory 26 (1997) 361–366.
[13] D. Gillies, Some theorems on n-person games, Ph.D. thesis, Princeton University, Department of Mathematics, 1953.
[14] M. Goemans, M. Skutella, Cooperative facility location games, J. Algorithms 50 (2004) 194–214, early version: SODA 2000, pp. 76–85.
[15] A. Gupta, A. Srinivasan, E. Tardos, Cost-sharing mechanisms for network design, in: Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, Cambridge, MA, 2004, pp. 139–150.
[16] J.P. Kahan, A. Rapoport, Theories of Coalition Formation, Lawrence Erlbaum Associates, 1984.
[17] R. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85–103.

[18] S. Ketchpel, Forming coalitions in the face of uncertain rewards, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), Seattle, WA, 1994, pp. 414–419.
[19] K. Larson, T. Sandholm, Anytime coalition structure generation: An average case study, J. Experiment. Theoret. AI 11 (2000) 1–20, early version: AGENTS 1999, pp. 40–47.
[20] E. Markakis, A. Saberi, On the core of the multicommodity flow game, in: Proceedings of the ACM Conference on Electronic Commerce (ACM-EC), San Diego, CA, 2003, pp. 93–97.
[21] A. Mas-Colell, M. Whinston, J.R. Green, Microeconomic Theory, Oxford University Press, Oxford, 1995.
[22] P. Milgrom, J. Roberts, Coalition-proofness and correlation with arbitrary communication possibilities, Games Econ. Behav. 17 (1996) 113–128.
[23] D. Moreno, J. Wooders, Coalition-proof equilibrium, Games Econ. Behav. 17 (1996) 80–112.
[24] A. Okada, A noncooperative coalitional bargaining game with random proposers, Games Econ. Behav. 16 (1996) 97–108.
[25] M.J. Osborne, A. Rubinstein, A Course in Game Theory, MIT Press, Cambridge, MA, 1994.
[26] M. Pal, E. Tardos, Group strategyproof mechanisms via primal-dual algorithms, in: Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS), 2003, pp. 584–593.
[27] C. Papadimitriou, Algorithms, games and the Internet, in: Proceedings of the Annual Symposium on Theory of Computing (STOC), 2001, pp. 749–753.
[28] I. Ray, Coalition-proof correlated equilibrium: A definition, Games Econ. Behav. 17 (1996) 56–79.
[29] T. Sandholm, K. Larson, M. Andersson, O. Shehory, F. Tohmé, Coalition structure generation with worst case guarantees, Artificial Intelligence 111 (1–2) (1999) 209–238, early version: AAAI 1998, pp. 46–53.
[30] T. Sandholm, V.R. Lesser, Coalitions among computationally bounded agents, Artificial Intelligence 94 (1) (1997) 99–137, special issue on Economic Principles of Multiagent Systems. Early version: IJCAI 1995, pp. 662–669.
[31] L.S. Shapley, On balanced sets and cores, Naval Research Logistics Quarterly 14 (1967) 453–460.
[32] O. Shehory, S. Kraus, A kernel-oriented model for coalition-formation in general environments: Implementation and results, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), Portland, OR, 1996, pp. 134–140.
[33] O. Shehory, S. Kraus, Methods for task allocation via agent coalition formation, Artificial Intelligence 101 (1–2) (1998) 165–200.
[34] F. Tohmé, T. Sandholm, Coalition formation processes with belief revision among bounded rational self-interested agents, J. Logic Comput. 9 (97–63) (1999) 1–23.
[35] W.J. van der Linden, A. Verbeek, Coalition formation: A game-theoretic approach, in: H.A.M. Wilke (Ed.), Coalition Formation, in: Advances in Psychology, vol. 24, North-Holland, Amsterdam, 1985, pp. 29–114.
[36] L.S. Wu, A dynamic theory for the class of games with nonempty cores, SIAM J. Appl. Math. 32 (1977) 328–338.
[37] G. Zlotkin, J.S. Rosenschein, Coalition, cryptography and stability: Mechanisms for coalition formation in task oriented domains, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), Seattle, WA, 1994, pp. 432–437.

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Artificial Intelligence 170 (2006) 620–642
Artificial Intelligence
www.elsevier.com/locate/artint

Is real-valued minimax pathological?

Mitja Luštrek [a,*], Matjaž Gams [a], Ivan Bratko [b]

[a] *Department of Intelligent Systems, Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia*
[b] *Faculty of Computer and Information Science, University of Ljubljana, Tržaška cesta 25, 1000 Ljubljana, Slovenia*

Received 4 January 2005; received in revised form 9 January 2006; accepted 15 January 2006

Available online 17 February 2006

Abstract

Deeper searches in game-playing programs relying on the minimax principle generally produce better results. Theoretical analyses, however, suggest that in many cases minimaxing amplifies the noise introduced by the heuristic function used to evaluate the leaves of the game tree, leading to what is known as pathological behavior, where deeper searches produce worse results. In most minimax models analyzed in previous research, positions' true values and sometimes also heuristic values were only losses and wins. In contrast to this, a model is proposed in this paper that uses real numbers for both true and heuristic values. This model did not behave pathologically in the experiments performed. The mechanism that causes deeper searches to produce better evaluations is explained. A comparison with chess is made, indicating that the model realistically reflects position evaluations in chess-playing programs. Conditions under which the pathology might appear in a real-value model are also examined. The essential difference between our real-value model and the common two-value model, which causes the pathology in the two-value model, is identified. Most previous research reports that the pathology tends to disappear when there are dependences between the values of sibling nodes in a game tree. In this paper, another explanation is presented which indicates that in the two-value models the error of the heuristic evaluation was not modeled realistically.

Keywords: Game playing; Minimax; Pathology; Game tree; Real value; Chess

1. Introduction

The minimax principle lies at the heart of almost every game-playing program. Programs typically choose the best move by searching the game tree, heuristically evaluating the leaves and then propagating their values to the root using the minimax principle. In practice, deeper searches generally produce better results. However, mathematical analysis indicates that under certain seemingly sensible conditions, the opposite is true: minimaxing amplifies the error of the heuristic evaluation [2,10]. This phenomenon is called the minimax pathology [10]. Nau [14] described this as "doing worse by working harder". Evidently, game trees of real games must be different from game trees used in the theoretical analyses in some way that eliminates the pathology. Several explanations of what property of game trees of real games might be responsible have been put forth: similarity of positions close to each other [3,11], presence of stabilizing game-tree nodes with reliable estimates [4,16] etc. And while these properties can be shown to eliminate

* Corresponding author.
E-mail addresses: mitja.lustrek@ijs.si (M. Luštrek), matjaz.gams@ijs.si (M. Gams), bratko@fri.uni-lj.si (I. Bratko).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2006.01.006

the pathology, the question to what extent they are actually present in game trees of real games remains. Also, the mechanism by which they achieve its goal is sometimes inadequately explained.

This paper attempts to address both of these problems: we model game trees in a way that we believe reflects well the properties of real games and we explain the mechanism that eliminates the pathology. In our game trees, positions close to each other are similar. But unlike most models used in previous research, in which positions' true values, and sometimes also heuristic values, could only be losses or wins, we use real numbers for both true and heuristic values. This makes it possible to model the similarity between a position's descendants in a more natural way, and enables a more direct comparison to game-playing programs, which typically use a sizable range of integer values. The model can be analyzed mathematically to explain the mechanism that makes minimax effective. Multiple types of error and different experimental settings, among them the approximation of real values by a limited number of discrete values, are used to determine when exactly is the pathology present in a real-value model. Some light is also shed on the reasons for pathological behavior of the two-value models used in previous research.

The paper is organized as follows. Section 2 is a short introduction to the minimax pathology with some historical overview. Section 3 presents our model of minimax. Section 4 gives a mathematical analysis of minimax with real-number values. Section 5 compares our model to a chess program. Section 6 explains when and how the pathology appears in a real-value model and examines the differences between real-value and two-value models. Section 7 concludes the paper and points out some areas where further research is needed.

2. Related work

The minimax pathology was discovered independently by Nau [10] and Beal [2]. It was later more thoroughly analyzed by many authors, among them Bratko and Gams [4], whose notation we adopt. In this section, we present a basic minimax model like the one by Beal and others.

A game tree consists of nodes representing game positions and edges representing moves. In the basic model, positions can be lost or won. Negamax notation is used in this section, i.e. nodes are marked as lost or won from the perspective of the side to move. Two situations are possible: a node has at least one lost descendant, in which case the node itself is won because one can always choose the move leading to the descendant lost for the opponent; or a node has only won descendants, in which case the node itself is lost because all moves lead to positions won for the opponent. This is shown in Fig. 1; losses are marked with "−" and wins with "+".

Virtually all research on the minimax pathology assumed game trees to have a uniform branching factor b. In the basic model, the value of each leaf is independent of other leaves' values. Let d be the depth of search and k_i the probability of a loss at ith ply. Plies are numbered upwards: 0 for the lowest ply of search and d for the root.

Since a node can only be lost if all of its descendants are won, the relation between the values of k at consecutive plies is governed by Eq. (1).

$$k_{i+1} = (1 - k_i)^b. \tag{1}$$

The goal is to calculate the probability of incorrectly evaluating the root given the probability of an incorrect evaluation at the lowest ply of search. Two types of evaluation error are possible: a loss can be mistaken for a win (false win) or a win for a loss (false loss). Let p_i and q_i be the probabilities of the respective types of error at ith ply. False wins occur in nodes where all descendants should be won, but at least one of them is a false loss. Therefore p_{i+1} can be calculated according to Eq. (2).

$$p_{i+1} = \frac{1}{k_{i+1}}(1 - k_i)^b\big(1 - (1 - q_i)^b\big) = 1 - (1 - q_i)^b. \tag{2}$$

False losses occur in nodes where some descendants should be lost, but all of those are false wins instead, while all won descendants retain their true values. Therefore q_{i+1} can be calculated according to Eq. (3).

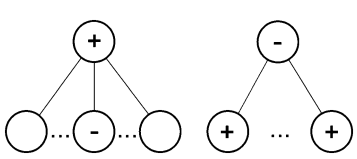

Fig. 1. Types of nodes.

$$q_{i+1} = \frac{1}{1-k_{i+1}} \sum_{j=1}^{b} \binom{b}{j} k_i^j (1-k_i)^{b-j} p_i^j (1-q_i)^{b-j}$$
$$= \frac{(k_i p_i + (1-k_i)(1-q_i))^b - ((1-k_i)(1-q_i))^b}{1-k_{i+1}}. \quad (3)$$

If only games where both sides have an equal chance of winning at the root of the game tree are considered, k_d must be 0.5 and Eq. (1) can be used to calculate k for other plies. Values for p_0 and q_0 have to be chosen and Eqs. (2) and (3) can be used to calculate p_d and q_d. If error at the root is defined as either $p_d k_d + q_d(1-k_d)$ [4] or $1/2(p_d + q_d)$ [16], it turns out that with increasing d, the error converges towards 0.5, rendering minimax useless.

First attempts to explain this phenomenon were made by Bratko and Gams [4] and Beal [3]. Both came to the conclusion that the reason minimax is effective in real games is that sibling nodes have similar values. Bratko and Gams argued that this property causes the formation of subtrees with reliably evaluated roots, which have a stabilizing effect. They did not verify whether this is indeed the case in real games. Beal showed that if a sufficient number of nodes that have all the descendants either lost or won are present in a game tree, the pathology disappears. He successfully verified his claim on king and pawn versus king chess endgame, but his results are not conclusive because such an endgame is not a typical chess situation. Nau [11,13] used a simple game designed for minimax analysis to show that a strong dependence between a node and its descendants eliminates the pathology. Pearl [16] suspected that real games do not have such a strong dependence. He argued that early terminations (also called blunders), which carry reliable evaluations, are the main reason for the elimination of the pathology. Both early terminations and subtrees with similar node values proposed by Bratko and Gams [4] result in relatively error-free nodes, so these two properties use basically the same mechanism to eliminate the pathology. Pearl's calculations show that the number of necessary early terminations in a game tree with $b = 2$ is 5%, but attempts by Luštrek and Gams [9] to verify this experimentally indicate that 20% is needed with $b = 2$ and even more with larger branching factors. Schrüfer [19] showed that if the error, particularly the probability of a false loss, is sufficiently low and certain additional conditions are satisfied, a game tree is not pathological. He did not investigate whether these conditions are present in real games. Althöfer [1] showed that in the trees that are pathological according to Schrüfer, the pathology persists even if some other back-up procedure is used instead of minimax.

Some researchers used models with multiple heuristic values, while they kept the true values restricted to losses and wins. Nau [11,13] used multiple heuristic values simply because they were a natural choice for the game he studied. In [12], he was concerned with proving that in certain circumstances, when the evaluation function has a limited number of possible values and the branching factor is large enough, the pathology is inevitable. Later [13] he extended the proof to multiple true values. Pearl [16] also considered multiple heuristic values and concluded that with regard to the pathology, they behave the same way as two values. Scheucher and Kaindl [18] were the first to consider multiple heuristic values crucial for the elimination of the pathology. They used them to design a heuristic evaluation function representative of what game-playing programs use. According to this function, not only were the values of the leaves of the game tree not independent of each other, it also implied that at lower plies of the game tree, positions where error is less likely are encountered more often than at higher plies. These two characteristics eliminated the pathology, but it remained unclear why the error should depend on the depth of search exactly the way they proposed.

A model with an arbitrary number of heuristic values and the same number of true values was already studied by Bratko and Gams [4]. They used completely independent leaf values and this model behaved similarly to the two-value version. Sadikov et al. [17] used multiple values in their analysis of king and rook versus king chess endgame. They managed to explain the pathology in backed-up position evaluations, but it is not known whether their conclusion applies to cases other than the endgame they studied. Some preliminary results of the research fully described in this paper were published by Luštrek [8].

3. A model with real-number values

There are many games where the final outcome is multivalued, typically expressed as an integer (Othello, bridge…). In such games, multivalued position values, both true and heuristic, are obviously needed. In games where the outcome can only be a loss, a win and perhaps a draw (chess, checkers…), multiple values might only seem necessary to express the uncertainty of the heuristic evaluation function, not as true values. However, even unlimited

resources to determine the value of a position cannot remove the uncertainty: in a losing position, the best one can do against a fallible and not fully known opponent is to evaluate moves in terms of the probability of a loss against this particular opponent. More importantly, in a winning position, multiple values make it possible to maintain a *direction* of play, gradually moving toward a win, not just maintaining a won position without achieving the final goal. Multiple values are necessary to differentiate between multiple winning (or losing, if only such are available) moves. Scheucher and Kaindl [18] already discussed the need for multiple values, although they were only concerned with heuristic values. They demonstrated on chess that a two-valued evaluation function performs poorly compared to a multivalued one. Given that multivalued position values are necessary for playing well, it is natural to also use multivalued true values. These values are true in the sense that they guide a program to play optimally. In our minimax model we use real values instead of multiple discrete values. This facilitates the mathematical explanation of why minimax works. The issue of granularity, i.e. how many distinct values should be used, is studied separately in Section 6.2. In particular, in Section 6.2 we address the question: Are the desirable properties of our real-valued model retained when real values are replaced by a limited number of discrete values?

3.1. Description of the model

In our model, the values of sibling nodes are distributed around the value of their parent. The rationale for such a distribution is that the descendants of a position are only one move away from it and are therefore unlikely to have a significantly different value. Dependence between the values of nodes and their descendants results in dependence between the values of leaves, which is a departure from the Section 2's assumption of independent leaf values. Negamax notation is no longer used, but the branching factor is still constant.

According to our model, a game tree is constructed from the root down. A so-called auxiliary value of 0 is assigned to the root and the auxiliary values of its descendants are distributed around it. The process is repeated recursively on each descendant until the maximum depth $d_{\max}$ required for a given experiment is reached. It should be noted that the auxiliary values generally do not respect the min/max rule. They serve only to establish the desired relation between the values of the leaves. Such a model is similar to Nau's incremental [11] or N-games [12] and to the model used by Scheucher and Kaindl [18], as well as to some earlier analyses of the alpha-beta algorithm [5,7,15].

The auxiliary values of the leaves of a tree with depth $d = d_{\max}$ are considered true values. True values at depths $d < d_{\max}$ are obtained by performing minimax starting with the true values of the leaves. In experiments with minimax search to depth d, static heuristic values are obtained by corrupting the true values at depth d with error representing the fallibility of the heuristic evaluation function. The error at the lowest ply of search, i.e. at depth d, is called static error. Backed-up heuristic values are obtained by performing minimax on the static heuristic values. The procedure is illustrated in Fig. 2.

The error is normally distributed noise. Normal distribution is chosen because this is the usual practice when no useful information on the error is available. Two types of error can be observed in a node: *position error*, which is the difference between the true and the heuristic value of the node, and *move error*, which is the probability of choosing a wrong move because of position error at the node's descendants.

The model has a number of parameters:

- b—branching factor,
- type of distribution of nodes' auxiliary values around the auxiliary value of their parent,
- σ_v—standard deviation of the above distribution,

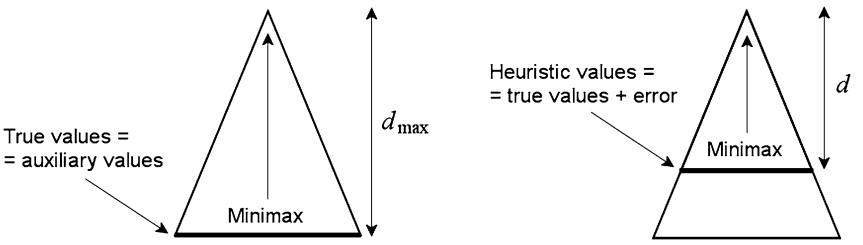

Fig. 2. Construction of game trees.

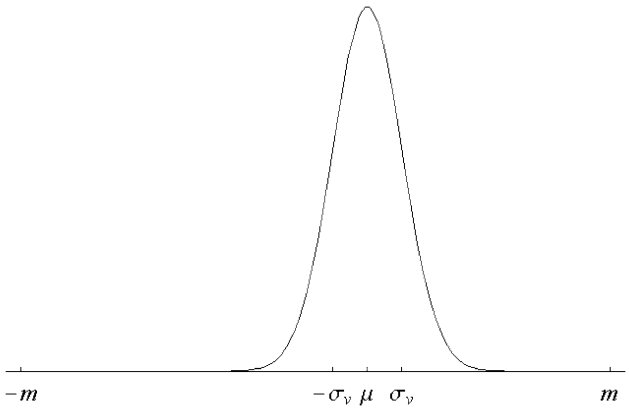

Fig. 3. Parameters of the model.

- $[-m, m]$—the interval within which nodes' values must lie,
- σ_e—standard deviation of static position error.

Some of the parameters are illustrated in Fig. 3; the curve represents probability density function of the distribution of sibling nodes' auxiliary values (in this case normal) and μ is the auxiliary value of their parent.

3.2. Experimental results

We investigate the behavior of minimax by randomly generating game trees and observing position and move error. The size of a game tree exceeds b^d, so computations with large branching factors or depths are very time-consuming. We conducted experiments with $b = 2$ and $b = 5$. The results were similar in all cases, so with the exception of the first experiment (Table 1 and Fig. 4), only the results with $b = 2$ are presented. Only depths of up to 10 were used, because observations at greater depths yield no additional insight.

Only relative values of σ_v, m and σ_e are important, so σ_v can be fixed to 1. In Table 1 and Fig. 4, results with normal distribution of nodes' values, $m = \infty$ and $\sigma_e = 0.2$ are presented. The parameters are chosen rather arbitrarily, but they will be examined later on: the value of each parameter will be varied while the others will be kept unchanged. The results are averaged over 10,000 game trees for $b = 2$ and 2,500 game trees for $b = 5$. For each tree, there are 10 repetitions with randomly generated noisy values. This will be the case in all experiments in the paper with the exception of Section 6.2. Average position error is defined as the sum of absolute differences between the true and the heuristic value of the root divided by the number of trees. Average move error is defined as the number of trees where a non-optimal move is chosen at the root divided by the number of all trees. Both position and move error at the root with respect to the depth of search are given in Table 1 and Fig. 4. Move error is more important because the choice

Table 1
Position and move error with the initial parameter settings

d	$b = 2$		$b = 5$	
	Position error	Move error	Position error	Move error
0	0.1599	–	0.1586	–
1	0.1588	0.0361	0.1545	0.1008
2	0.1549	0.0356	0.1457	0.0959
3	0.1521	0.0350	0.1403	0.0938
4	0.1501	0.0339	0.1344	0.0908
5	0.1478	0.0340	0.1283	0.0852
6	0.1460	0.0335	0.1242	0.0809
7	0.1438	0.0330	0.1177	0.0789
8	0.1415	0.0325	0.1138	0.0764
9	0.1387	0.0315	0.1108	0.0749
10	0.1361	0.0314	0.1051	0.0746

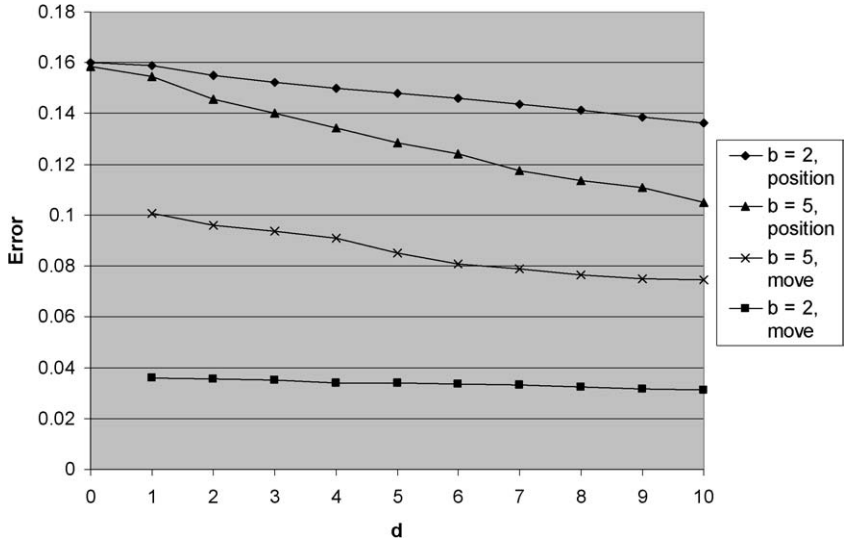

Fig. 4. Position and move error with the initial parameter settings.

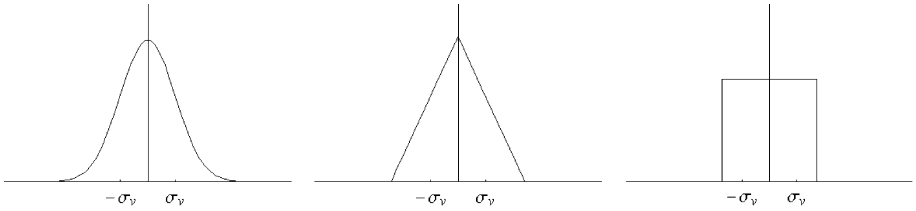

Fig. 5. Types of distribution of nodes' auxiliary values around the auxiliary value of their parent.

Table 2
Move error with different distributions of nodes' auxiliary values

d	Normal	Triangular	Uniform
1	0.0361	0.0366	0.0359
2	0.0356	0.0355	0.0352
3	0.0350	0.0351	0.0350
4	0.0339	0.0347	0.0345
5	0.0340	0.0338	0.0338
6	0.0335	0.0340	0.0339
7	0.0330	0.0332	0.0336
8	0.0325	0.0332	0.0331
9	0.0315	0.0317	0.0326
10	0.0314	0.0315	0.0314

of move is the purpose of minimax. Also, the behavior of both types of error in the experiments was similar, so only move error will be presented in the rest of the section.

As can be seen in Table 1 and Fig. 4, increased depth of search is generally beneficial.

Table 2 and Fig. 6 show how the choice of *the distribution of nodes' auxiliary values* affects move error at the root with respect to the depth of search. Normal, triangular and uniform distributions were tested to investigate the effect of varying bias towards the mean—they are shown in Fig. 5. The other parameters remain unchanged: $b = 2$, $m = \infty$ and $\sigma_e = 0.2$.

As can be seen in Table 2 and Fig. 6, the choice of the distribution has very little effect on the behavior of move error. Apparently it matters only that nodes' values are similar to the value of their parent, not the exact shape of the similarity.

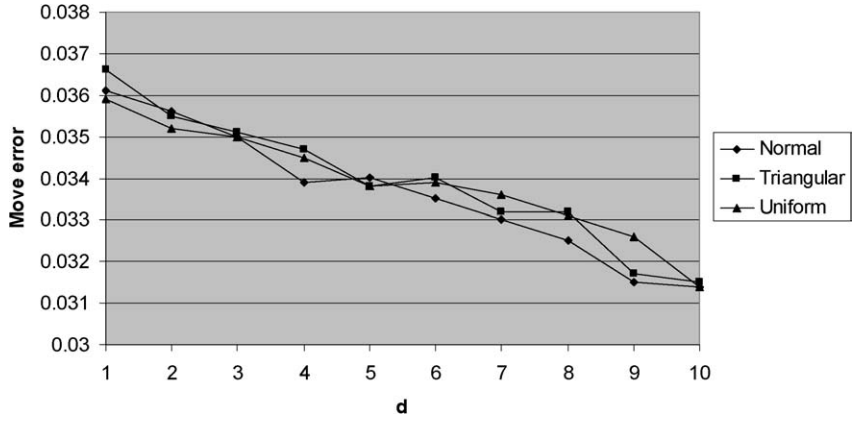

Fig. 6. Move error with different distributions of nodes' auxiliary values.

Table 3
Move error with different values of m

d	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=\infty$
1	0.2040	0.0820	0.0500	0.0383	0.0371	0.0361
2	0.1995	0.0791	0.0482	0.0386	0.0371	0.0356
3	0.1952	0.0779	0.0478	0.0388	0.0365	0.0350
4	0.1928	0.0761	0.0472	0.0367	0.0365	0.0339
5	0.1886	0.0748	0.0458	0.0369	0.0357	0.0340
6	0.1861	0.0742	0.0462	0.0364	0.0352	0.0335
7	0.1837	0.0725	0.0436	0.0351	0.0343	0.0330
8	0.1797	0.0691	0.0437	0.0348	0.0336	0.0325
9	0.1776	0.0704	0.0440	0.0352	0.0334	0.0315
10	0.1766	0.0677	0.0430	0.0342	0.0327	0.0314

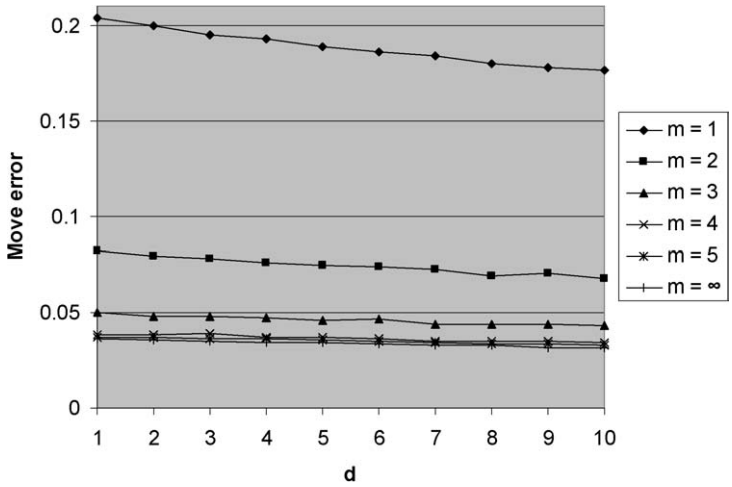

Fig. 7. Move error with different values of m.

The initial setting of $m=\infty$ means that nodes' values are unconstrained, i.e. that however good or bad a position is, it can always get better or worse respectively. In real games, position values are not unbounded: there is checkmate in chess, maximum or minimum number of pieces in Othello etc. In our model, the bound is implemented by simply setting the nodes' values that fall outside the $[-m, m]$ interval to $-m$ or m. Table 3 and Fig. 7 show how *the interval within which the nodes' values must lie* affects move error at the root with respect to the depth of search. The other

Table 4
Move error with different values of σ_e

d	$\sigma_e = 1$	$\sigma_e = 0.5$	$\sigma_e = 0.2$	$\sigma_e = 0.1$
1	0.1682	0.0914	0.0361	0.0191
2	0.1619	0.0873	0.0356	0.0191
3	0.1532	0.0847	0.0350	0.0191
4	0.1455	0.0819	0.0339	0.0185
5	0.1373	0.0788	0.0340	0.0186
6	0.1308	0.0756	0.0335	0.0185
7	0.1224	0.0735	0.0330	0.0187
8	0.1172	0.0723	0.0325	0.0182
9	0.1095	0.0685	0.0315	0.0184
10	0.1019	0.0661	0.0314	0.0180

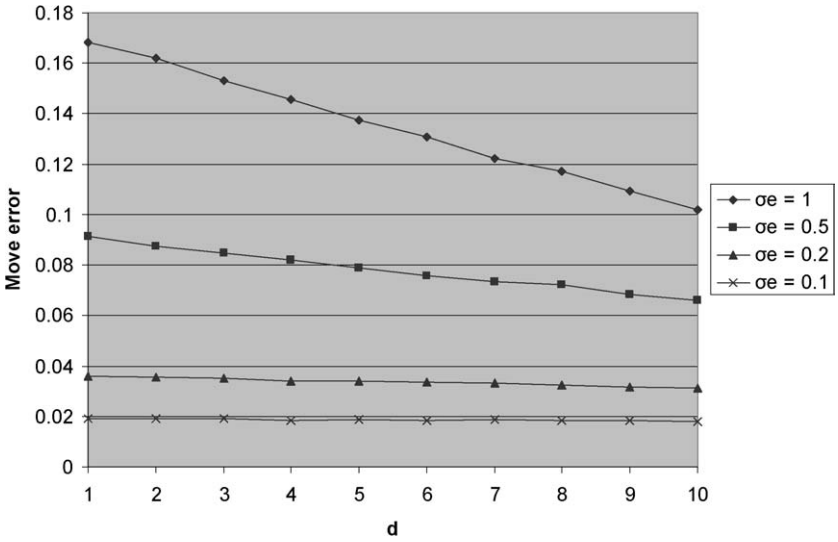

Fig. 8. Move error with different values of σ_e.

parameters remain unchanged: $b = 2$, distribution of nodes' values (when the values are unconstrained) is normal and $\sigma_e = 0.2$.

As can be seen in Table 3 and Fig. 7, the pathology is not present regardless of the value of m. However, m affects the magnitude of move error: when m is small, move error is large because the nodes' values are relatively closer to each other, which makes position error more likely to change their order. When m increases to the relatively small value of 4, results are very similar to those with $m = \infty$, and when it increases to 5, they are virtually identical.

Table 4 and Fig. 8 show how *standard deviation of static position error* affects move error at the root with respect to the depth of search. The other parameters remain unchanged: $b = 2$, distribution of nodes' values is normal and $m = \infty$.

As can be seen in Table 4 and Fig. 8, increased depth of search is beneficial for all σ_e. The benefit is less pronounced when σ_e is small, but this is to be expected since game tree search would not be beneficial at all if a perfect evaluation function were available.

We can conclude that in our model, minimax is not pathological for a wide range of parameter settings.

4. Mathematical explanation

In this section we attempt a mathematical explanation of the experimental observations in Section 3. For simplicity of the analysis we will assume that the difference between the true values of sibling nodes is always 1, which leads

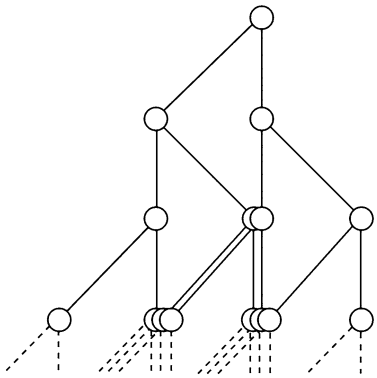

Fig. 9. Model used in the mathematical explanation.

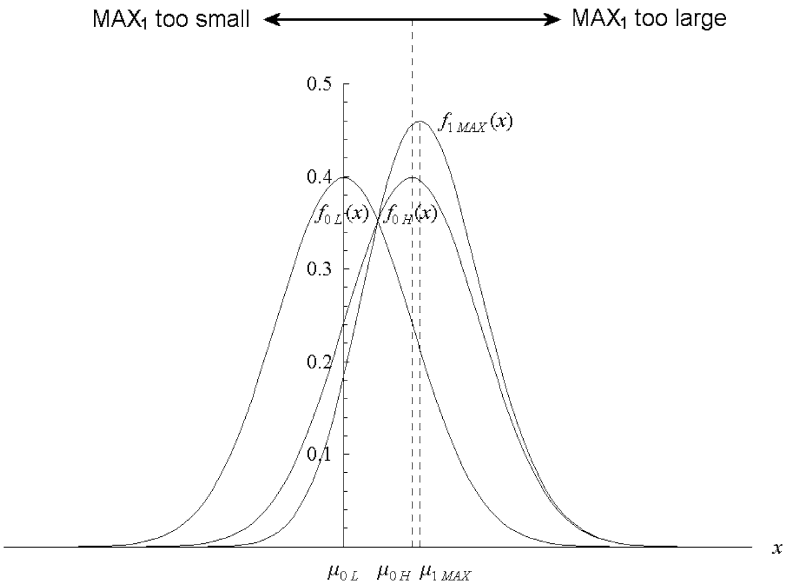

Fig. 10. Probability density functions of heuristic values of a pair of sibling nodes at ply 0 and their parent.

to the model sketched in Fig. 9. Hopefully this simplification does not affect qualitative conclusions of the analysis. Reasons why this should be the case are discussed later in this section. Also, only $b = 2$ is considered.

A node at ply $i + 1$ has two descendants whose heuristic values are random variables L_i (lower value) and H_i (higher value) with means $\mu_{i\,L}$ and $\mu_{i\,H}$. Due to normally distributed error, static heuristic values of sibling nodes are distributed normally with means μ_{0L} and μ_{0H} and standard deviation σ_e. Their means are also the nodes' true values. Their probability density functions are given in Eqs. (4).

$$f_{0L}(x) = \frac{1}{\sigma_e\sqrt{2\pi}} \mathrm{e}^{\frac{-(x-\mu_{0L})^2}{2\sigma_e^2}}, \qquad f_{0H}(x) = \frac{1}{\sigma_e\sqrt{2\pi}} \mathrm{e}^{\frac{-(x-\mu_{0H})^2}{2\sigma_e^2}}. \tag{4}$$

At a max ply $i + 1$, the value of a node is a random variable $MAX_{i+1} = \max(L_i, H_i)$ and its probability density function is calculated according to Eq. (5).

$$f_{i+1MAX}(x) = f_{iH}(x)P(L < x) + f_{iL}(x)P(H < x) = f_{iH}(x) \int_{-\infty}^{x} f_{iL}(l)\,\mathrm{d}l + f_{iL}(x) \int_{-\infty}^{x} f_{iH}(h)\,\mathrm{d}h. \tag{5}$$

Probability density functions of L_0, H_0 and MAX_1 are shown in Fig. 10. Unlike L_0 and H_0, MAX_1 is not distributed normally. Its probability density function was obtained by numerical integration.

As can be seen in Fig. 10, the curve of the parent is narrower than the curves of its descendents, meaning that the variance of the parent is smaller. Because the difference between the true values of sibling nodes is constant, the difference between the means of the heuristic values of sibling nodes is constant as well. Therefore a smaller variance means a smaller probability that the lower-valued sibling becomes, due to position error, the higher-valued sibling, i.e. a smaller move error. If the mean of the parent were equal to the mean of the higher descendant, a smaller variance would also mean a smaller position error. Since the mean of the parent is somewhat larger, this is not necessarily the case and will be examined later on.

Let us first explain what causes the reduction of variance. If there were no interaction between the descendants, f_{i+1MAX} would be equal to f_{iH}. Since this is not true, two cases must be considered. When MAX_{i+1} is too large ($MAX_{i+1} > \mu_{iH}$), the most likely reason is that H_i is too large ($H_i > \mu_{iH}$). The less likely reason is that L_i is much too large ($L_i > \mu_{iH}$). This is marked in Fig. 10 on the top. When MAX_{i+1} is too small ($MAX_{i+1} < \mu_{iH}$), this is caused by H_i being too small ($H_i < \mu_{iH}$), but it can be compensated by L_i being larger than H_i ($H_i < L_i < \mu_{iH}$). The corrective effect of L_i in the latter case is greater than the disturbing effect in the former case, so the impact of L_i it is more likely to be beneficial than harmful.

At a min ply, the probability density function is calculated according to Eq. (6).

$$f_{i+1MIN}(x) = f_{iL}(x)P(H > x) + f_{iH}(x)P(L > x) = f_{iL}(x)\int_x^\infty f_{iH}(h)\,\mathrm{d}h + f_{iH}(x)\int_x^\infty f_{iL}(l)\,\mathrm{d}l. \tag{6}$$

Probability density functions at ply 0 are given by Eqs. (4), so probability density function at any ply can be calculated by repeatedly using Eqs. (5) and (6). This is shown in Fig. 11; $\mu_{iH} - \mu_{iL} = 1$ for all i.

As can be seen in Fig. 11, the higher a ply, the narrower the curve of the probability density function of a node's heuristic backed-up value. This means that the variance of the heuristic values and consequently move error decrease with the depth of search. True values in all the cases in Fig. 11 are 0, so it can also be seen that heuristic values exhibit a slight positive bias, most of which is accrued at the lowest ply of search. This is consistent with observations by Sadikov et al. [17].

Let ME_{i+1} be move error at ply $i+1$. An erroneous move at ply $i+1$ is chosen when the values of a pair of sibling nodes at ply i are switched, i.e. $L_i > H_i$, so ME_{i+1} is calculated according to Eq. (7).

$$ME_{i+1} = P(L_i > H_i) = \int_{-\infty}^{\infty} f_{iH}(h)\left(\int_h^\infty f_{iL}(l)\,\mathrm{d}l\right)\mathrm{d}h. \tag{7}$$

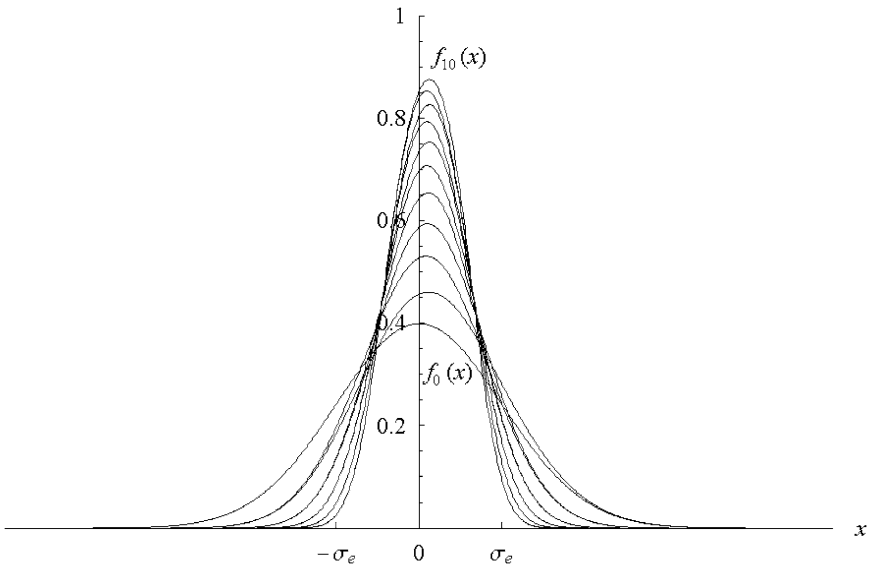

Fig. 11. Probability density functions of the values of nodes at plies 0 through 10.

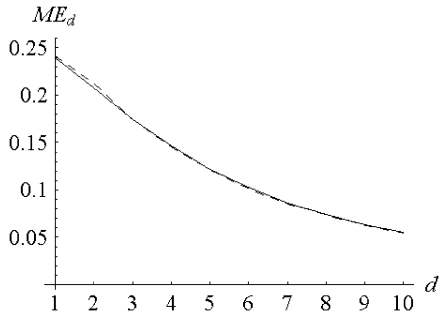

Fig. 12. Move error at the root as a function of the depth of search (dashed line shows Monte Carlo experimental results).

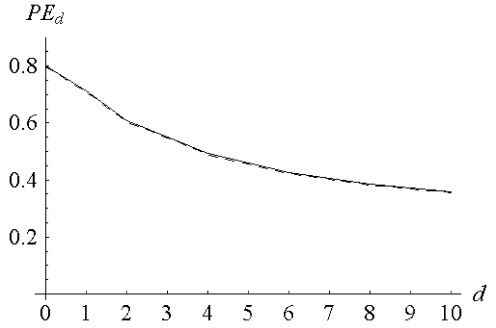

Fig. 13. Position error at the root as a function of the depth of search (dashed line shows Monte Carlo experimental results).

How move error changes with increasing depth of search is shown in Fig. 12; ME_d is move error at the root when the depth of search is d, $\mu_{i\,H} - \mu_{i\,L} = 1$ for all i and $\sigma_e = 1$. The solid line results from calculations according to Eq. (7). The dashed line shows the results of Monte Carlo experiments using the model of this section (the difference between the true values of sibling nodes always 1). They serve to verify the correspondence between the mathematical analysis and random experiments.

As can be seen in Fig. 12, increased depth of search reduces move error at the root.

Let t_i be the true value of a node at ply i and $f_{i\,E}$ probability density function of its heuristic (erroneous) value E_i. Position error PE_i is the average absolute difference between t_i and E_i and is calculated according to Eq. (8).

$$PE_i = \int_{-\infty}^{\infty} |t_i - x| f_{i\,E}(x)\,\mathrm{d}x. \tag{8}$$

How position error changes with increasing depth of search is shown in Fig. 13; PE_d is position error at the root when the depth of search is d, $\mu_{i\,H} - \mu_{i\,L} = 1$ for all i and $\sigma_e = 1$. The solid line results from calculations according to Eq. (8). The dashed line again results from Monte Carlo experiments.

As can be seen in Fig. 13, increased depth of search reduces position error at the root, which is in accordance with the experimental results in Section 3. This means that the bias observed in Fig. 10 and Fig. 11 is not large enough to cause position error to behave pathologically, probably because alternating max and min plies cause alternating positive and negative bias. To verify this hypothesis, behavior of move and position error in a tree with only max plies was investigated. The results are shown in Fig. 14; $\mu_{i\,H} - \mu_{i\,L} = 1$ for all i and $\sigma_e = 1$.

As can be seen in Fig. 14, move error is still not pathological, while position error behaves pathologically for $d > 3$. This result is similar to what Sadikov et al. [17] observed in their experiments on king and rook versus king chess endgame, although their trees consisted of both max and min plies. The reason for this discrepancy is not clear.

In the analysis presented in this section, a constant difference between the true values of sibling nodes is assumed. However, in game trees of real games as well as game trees used in the experiments in Section 3, the difference between the true values of sibling nodes is not constant. The simplification in the analysis is not as problematic as it may appear, though. This is because we investigate the search of a *single* game tree to different depths, therefore no

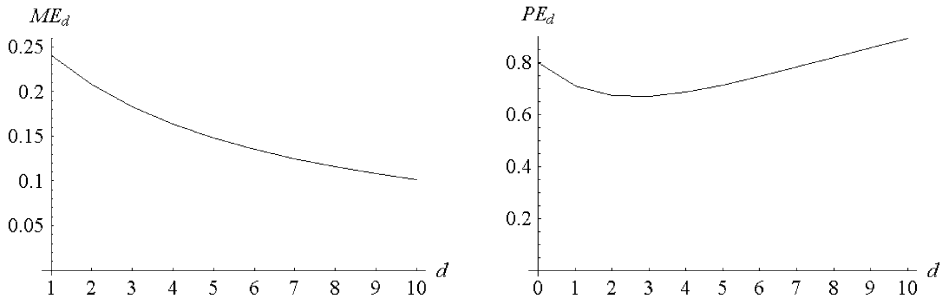

Fig. 14. Move error (left) and position error (right) at the root as a function of the depth of search in a tree with only max plies.

matter how the difference between the true values of sibling nodes varies within the tree, the variation is the same in searches to all depths. Hence the difference affects the error in a similar fashion in all searches and consequently does not affect the pathology.

5. Verification in a chess program

Whether the results of Sections 3 and 4 have bearing on real games can be verified by comparing the model proposed in Section 3 to real games. However, it is difficult to obtain true values of game-tree nodes of a sufficiently complex game. What can and often is done is to use values returned by a game-playing program. The game chosen is chess and the program is Crafty [6]. With ELO rating of 2,616, it is the highest-rated open-source chess program according to SSDF [20]. Being open-source is a requirement because the program needed to be modified for the experiment.

The comparison between the model and Crafty should be based on the values of the leaves. Distributions of leaf values are difficult to compare in practice, though, so the comparison is performed on the relations between the static values of nodes and their descendants in Crafty and auxiliary values of nodes and their descendants in our model. The values of the leaves are, after all, determined by this relation in both cases.

One tenth of 4.5 million positions visited in the course of a game played by Crafty were randomly chosen. All the positions generated during game-tree search were considered, not only positions actually occurring in the game. The differences between the static value of each node and the static values of all of its descendants were computed. Determining static values did not involve quiescence search. The results were discretized and the number of cases in each interval counted. In Fig. 15, the results are shown as the number of cases where the difference lies within an interval; the lower and upper 1% of the cases are omitted, otherwise the interesting interval could not be seen clearly since the extremes are -22.78 and 20.58.

As can be seen in Fig. 15, most nodes' static values lie close to the static value of their parent. Their distribution resembles normal. This relation corresponds reasonably well to the relation between the auxiliary values of nodes in our model, where the default distribution is also normal. Since Section 3 shows that the exact type of distribution is not crucial anyway, we feel that the results from Crafty can be considered a confirmation of the relevance of the model presented in Section 3 to real games.

6. When does pathology occur

If a two-value model can be pathological, as shown in Section 2, should not a real-value model exhibit pathology under certain conditions as well? They both aspire to model the error behavior of minimax in game-playing programs. The presence of the pathology in one and the complete absence in another would be an indication that one of them lacks some fundamental property. In this section we look into this apparent contradiction between the two models and search for conditions under which pathology would appear in a real-value model.

There are three main differences between the two-value model described in Section 2 and the real-value model described in Section 3:

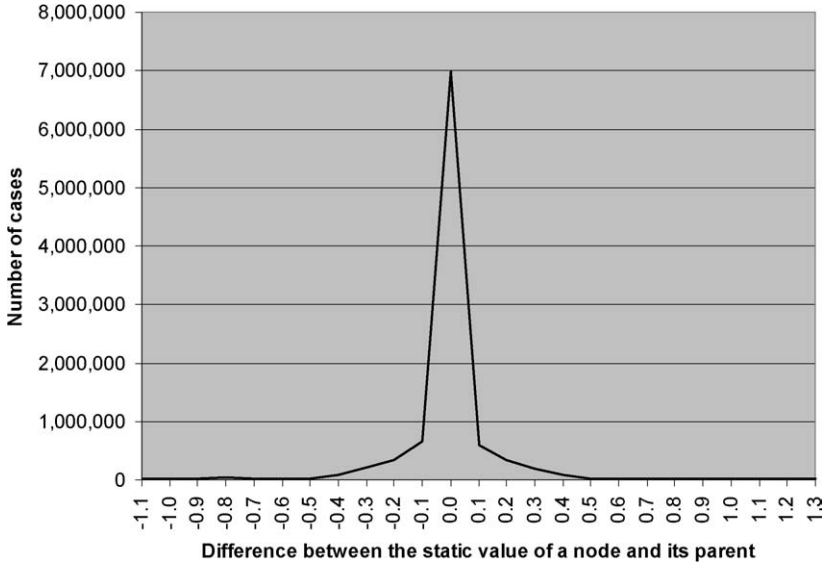

Fig. 15. Distribution of the differences between the static value of a node and its parent.

1. In the two-value model, the value of each leaf of the game tree is independent of all other leaves' values; in the real-value model, the leaves' values are not independent due to the nodes' auxiliary values being distributed around the auxiliary value of their parent.
2. In the two-value model, the error is measured as the probability of mistaking a loss for a win or vice versa and is set to the same value at the lowest ply of search regardless of the depth of search; in the real-value model, the error is measured as position error whose probability distribution is always the same at the lowest ply of search regardless of the depth of search, and move error for which this is not guaranteed as we show later in this section.
3. And of course one model uses two values and the other real values.

6.1. Independent-value model

If the first difference is removed, a game tree is constructed by simply setting the leaves' values independently of each other instead of using auxiliary intermediate values; the rest of the procedure follows the description in Section 3. We call such a model *independent-value* model as opposed to *dependent-value* model, which is the one described in Section 3. Table 5 and Fig. 16 show position and move error at the root with respect to the depth of search in the independent-value model; $b = 2$, distribution of leaves' values (this time around 0 instead of their parents' auxiliary values) is normal, $m = \infty$ and $\sigma_e = 0.2$.

Table 5
Position and move error at the root in the independent-value model

d	Position error	Move error
0	0.1598	–
1	0.1472	0.2952
2	0.1233	0.2916
3	0.1153	0.2863
4	0.1041	0.2824
5	0.0997	0.2744
6	0.0944	0.2720
7	0.0918	0.2686
8	0.0895	0.2635
9	0.0876	0.2578
10	0.0862	0.2577

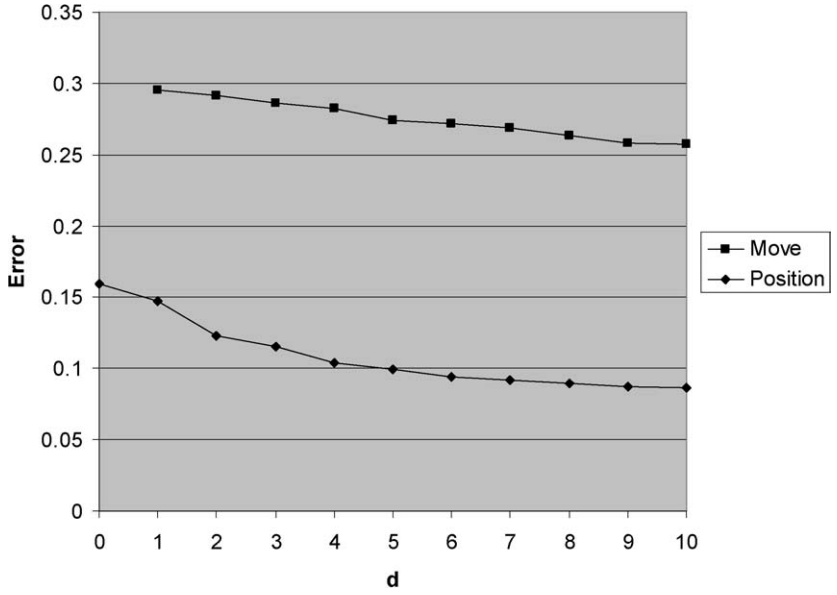

Fig. 16. Position and move error at the root in the independent-value model.

As can be seen in Table 5 and Fig. 16, the independent-value model with the chosen set of parameters is not pathological. Experiments with several different sets of parameters also yielded no pathology. Considering that models with independent values such as the one described in Section 2 are known to be pathological, this result is on one hand somewhat surprising. On the other hand, though, the explanation of Section 4 also applies to the independent-value model. Apparently two-value models are sufficiently different from real-value models that the pathology in one does not imply the pathology in the other, even if both are independent-valued.

6.2. Granularity

In our models, we used real numbers as both true and heuristic values, so in principle there could be an infinite number of them. Since digital computers do not use real values in the true mathematical sense, in our simulations these real numbers were approximated by high-density floating-point numbers. But we know that game-playing programs usually use as heuristic values a much smaller set of discrete values, most often represented by a range of integers. Therefore real values are not completely realistic for modeling typical game playing programs. We will now investigate whether this difference affects our main findings.

To see whether the issue of granularity affects the pathology, the number of possible heuristic values, or grains, in our models was restricted. The granularity of true values was set to be equal to the granularity of heuristic values, otherwise errors would occur even with the best possible evaluation function, simply because it would be too coarse-grained to differentiate between similar true values. The effect of granularity turned out to be strongly tied to the branching factor. Table 6 and Fig. 17 show the number of grains needed to avoid the pathology with respect to the branching factor in the dependent- and the independent-value model; the distribution of nodes' values in the dependent-value model is normal and in the independent-value model uniform, $d_{\max} = 6$, $m = \infty$ and $\sigma_e = 0.2$.

As can be seen in Table 6 and Fig. 17, the dependent-value model is quite robust in the sense that even with a low granularity, it is not pathological. Evaluation functions of modern chess programs have ranges of several thousand values, so these programs are not likely to be pathological due to low granularity. The independent-value model is much more sensitive to the granularity: the granularity required to avoid the pathology $g(b)$ seems to be exponential in the branching factor. Eq. (9) gives the least-square-error approximation of the experimental data.

$$g(b) = 7.5084 \times 1.3208^b - 0.5169. \tag{9}$$

Table 6
Required granularity to avoid the pathology in the dependent- and the independent-value model

b	Dependent-value model	Independent-value model
2	10	13
3	15	17
4	19	22
5	19	29
6	19	39
7	19	52
8	19	71
9	19	90
10	19	121

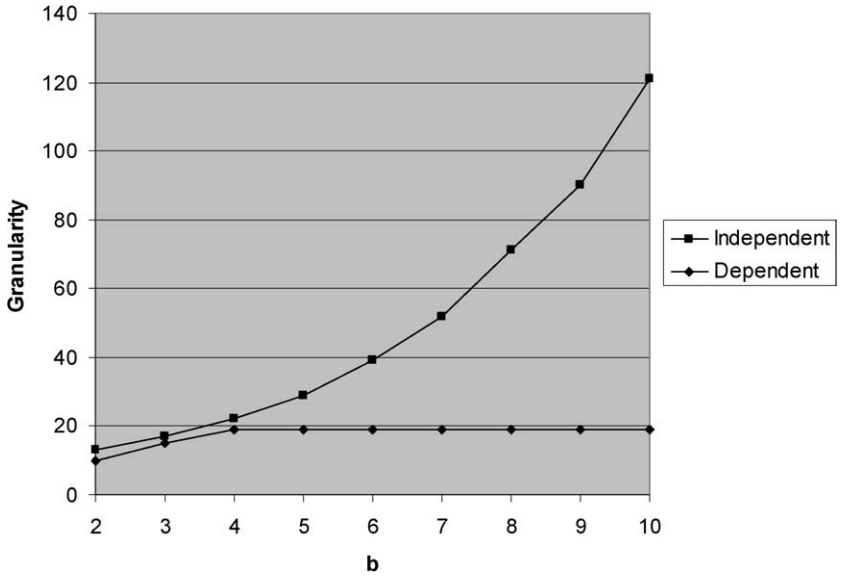

Fig. 17. Required granularity to avoid the pathology in the dependent- and the independent-value model.

The average branching factor of chess is estimated to be between 30 and 40. $g(30) = 31{,}670$ and $g(40) = 511{,}706$, so a game with a chess-like branching factor, typical chess program granularity and independent leaf values would probably be pathological.

This result is similar to what Nau [12] predicted: if the heuristic evaluation function returns its minimum and maximum values with a non-zero probability and the branching factor is large enough, the pathology will ensue. With the introduction of the granularity, every value in our model is indeed returned with a non-zero probability. Scheucher and Kaindl [18] also investigated the issue of granularity. Their conclusion was that it has no effect as long as it is not too small for the evaluation function to express all the knowledge of the game. In our experiments this was never the case, because the granularity of the heuristic values was equal to the granularity of the true values, but the pathology still appeared. Scheucher and Kaindl's did not encounter it because their work was based on the assumption that a simple chess evaluation function requires 84 distinct values, and since their model was dependent-valued, our experiments suggest that much fewer distinct values are required for the pathology to appear.

6.3. Constant static move error

In the experiments up to this point, whenever searches to different depths were performed, static position error was always constant. This is a reasonable assumption, since position error directly models the fallibility of the heuristic

Table 7
Static move error in the dependent- and the independent-value model

d	Dependent-value model	Independent-value model
1	0.0369	0.2952
2	0.0373	0.2602
3	0.0375	0.2249
4	0.0376	0.1946
5	0.0384	0.1636
6	0.0396	0.1375
7	0.0408	0.1130
8	0.0432	0.0936
9	0.0486	0.0760
10	0.0628	0.0628

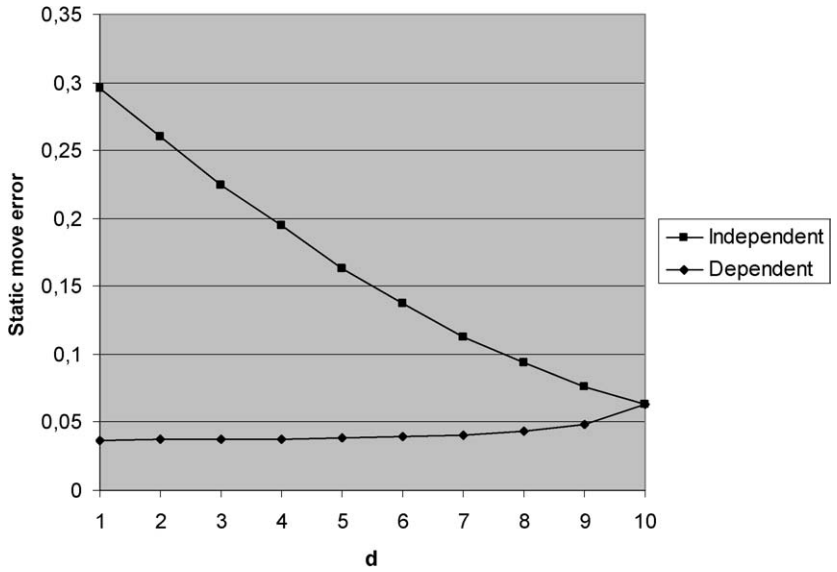

Fig. 18. Static move error in the dependent- and the independent-value model.

evaluation function and it is generally accepted that the heuristic evaluation function performs equally well at all plies within the horizon of a single search. However, it is also not an unreasonable assumption that move error at the lowest ply of search, or static move error, should be constant.

Table 7 and Fig. 18 show static move error with respect to the depth of search in the dependent- and the independent-value model; $b = 2$, distribution of nodes' values is normal, $m = \infty$ and $\sigma_e = 0.2$.

Perhaps surprisingly, as can be seen in Table 7 and Fig. 18, static move error is not constant. In the dependent-value model, it increases with the depth of search, while in the independent-value model, it decreases with the depth of search. Differences in static move error when searching to different depths are particularly large in the independent-value model.

Why static move error changes with the depth of search can be explained by observing the average difference between the true values of sibling nodes at successive plies. Table 8 and Fig. 19 show the results for the dependent- and the independent-value model; $b = 2$, distribution of nodes' values is normal, $m = \infty$, $\sigma_e = 0.2$ and $d_{\max} = 10$.

As can be seen in Table 8 and Fig. 19, the behavior of the average difference between the true values of sibling nodes is the opposite of the behavior of static move error (shown in Table 7 and Fig. 18). This is not surprising: it has already been observed that small differences between backed-up values of sibling nodes cause large move errors and vice versa.

Table 8
The average difference between the true values of sibling nodes at successive plies in a game tree with $d_{\max} = 10$ in the dependent- and the independent-value model

Ply	Dependent-value model	Independent-value model
9	1.9313	0.1711
8	1.9191	0.2132
7	1.9040	0.2622
6	1.8925	0.3232
5	1.8763	0.4016
4	1.8231	0.4913
3	1.7580	0.6108
2	1.6464	0.7467
1	1.4638	0.9297
0	1.1286	1.1286

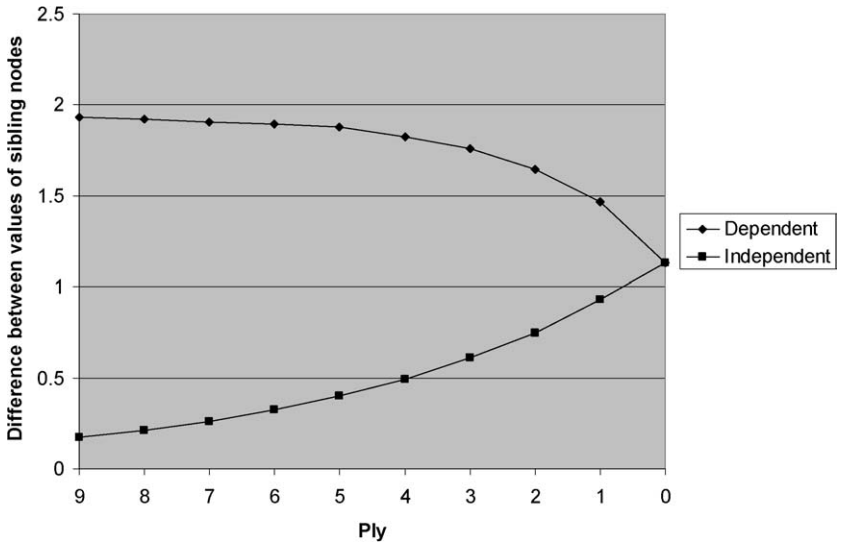

Fig. 19. The average difference between the true values of sibling nodes at successive plies in a game tree with $d_{\max} = 10$ in the dependent- and the independent-value model.

Why true backed-up values of sibling nodes in the dependent-value model are statistically further apart at higher plies can be explained by analyzing the relation between the auxiliary values of a pair of sibling nodes and the corresponding true backed-up values. Consider a pair of non-leaf sibling nodes in a dependent-value game tree. Let a_L and a_H be the auxiliary values of these sibling nodes, so that $a_L \leqslant a_H$. Let L^* and H^* be the corresponding true backed-up values; they are random variables that acquire values from the populations of the possible subtrees rooted in the two sibling nodes. The means of L^* and H^* are μ_{L^*} and μ_{H^*}. Both subtrees rooted in the two sibling nodes are constructed by the same random mechanism starting with the root values a_L and a_H, therefore $\mu_{H^*} - \mu_{L^*} = a_H - a_L$. Consider the difference $D = H^* - L^*$. The expected value of D, μ_D, is equal to $\mu_{H^*} - \mu_{L^*} = a_H - a_L$. This follows from the fact that the expected value of the difference between two random variables is equal to the difference between the expected values of the two variables. This property is demonstrated in Eq. (10), where X and Y are two independent random variables whose probability density functions are f_X and f_Y.

$$\mu_{X-Y} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - y) f_X(x) f_Y(y) \, \mathrm{d}x \, \mathrm{d}y$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x f_X(x) f_Y(y)\,\mathrm{d}x\,\mathrm{d}y - \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} y f_X(x) f_Y(y)\,\mathrm{d}x\,\mathrm{d}y$$

$$= \int_{-\infty}^{\infty} x f_X(x)\left(\int_{-\infty}^{\infty} f_Y(y)\,\mathrm{d}y\right)\mathrm{d}x - \int_{-\infty}^{\infty} y f_Y(y)\left(\int_{-\infty}^{\infty} f_X(x)\,\mathrm{d}x\right)\mathrm{d}y$$

$$= \int_{-\infty}^{\infty} x f_X(x) 1\,\mathrm{d}x - \int_{-\infty}^{\infty} y f_Y(y) 1\,\mathrm{d}y = \mu_X - \mu_Y. \tag{10}$$

Regarding the probability of move error, the parameter that matters is the absolute difference between the true values of the two sibling nodes: $|D| = |H^* - L^*|$. The expected value of $|D|$ is greater than μ_D which is easily seen from Eq. (11).

$$\mu_D = \int_{-\infty}^{\infty} x f_D(x)\,\mathrm{d}x \leqslant \int_{-\infty}^{\infty} |x| f_D(x)\,\mathrm{d}x = \mu_{|D|}. \tag{11}$$

So for auxiliary values of any pair of sibling nodes, the corresponding true backed-up values are further apart on average. This is also illustrated in Fig. 20.

Auxiliary values at all plies are distributed equally, so the average difference between the auxiliary values of a pair of sibling nodes is the same for all plies. Auxiliary values at ply 0 are also true values, therefore backed-up true values at ply 1 and higher are further apart on average than true values at ply 0. What makes $\mu_{|D|}$ larger at each successive ply is that the enlarging effect is cumulative.

Why true backed-up values of sibling nodes in the independent-value model are closer to each other at higher plies can be explained as follows. The values of the leaves are distributed within an interval. Applying maximum to groups of them results in values concentrated in a smaller interval. Applying minimum to groups of values from the new interval results in values concentrated in an even smaller interval, etc. This is illustrated in Fig. 21.

The effect of the average difference between the values of sibling nodes on static move error can be compensated by adjusting static position error to achieve constant static move error: in the dependent-value model, it has to be increased in shallower searches, while in the independent-value model, it has to be decreased in shallower searches. Table 9 and Fig. 22 show move error at the root with respect to the depth of search in the dependent- and the independent-value model with constant static move error; $b = 2$, distribution of nodes' values is normal and $m = \infty$.

As can be seen in Table 9 and Fig. 22, constant static move error strengthens the benefits of minimax in the dependent-value model, but produces the pathology in the independent-value model. This is consistent with previous

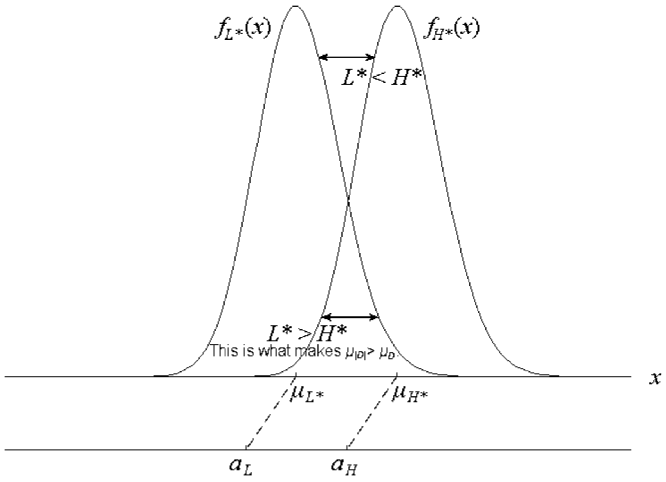

Fig. 20. Nodes' true backed-up values higher up in a tree are further apart in the dependent-value model.

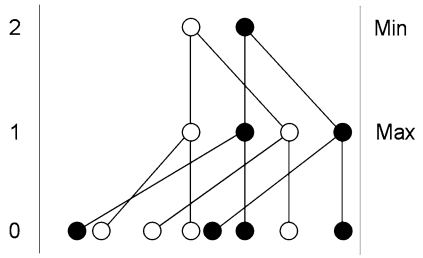

Fig. 21. Explanation for nodes' backed-up values higher up in a tree being closer to each other in the independent-value model.

Table 9
Move error at the root in the dependent- and the independent-value model with constant static move error

d	Dependent-value model	Independent-value model
1	0.0628	0.0628
2	0.0614	0.0734
3	0.0605	0.0914
4	0.0599	0.1069
5	0.0557	0.1303
6	0.0523	0.1501
7	0.0494	0.1756
8	0.0476	0.2064
9	0.0418	0.2317
10	0.0314	0.2577

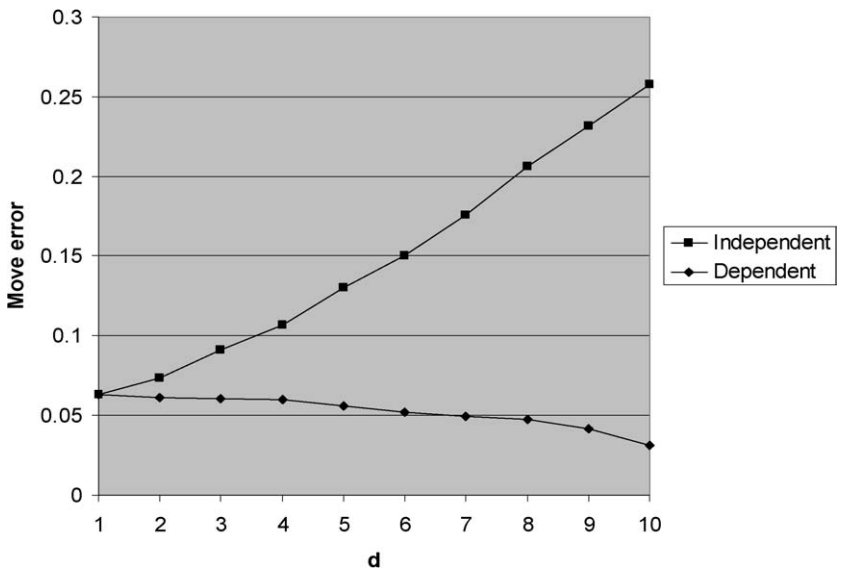

Fig. 22. Move error at the root in the dependent- and the independent-value model with constant static move error.

research by Beal [3], Bratko and Gams [4], and Nau [11,12], concerned with two-value models. But it should be noted that in two-value models, error is defined somewhat differently from move error in our real-value model.

Table 10
Two-value error at the root and at the lowest ply of search in the dependent- and the independent-value model

d	Dependent-value model		Independent-value model	
	Root	Static	Root	Static
0	0.0486	0.0486	0.2072	0.2072
1	0.0469	0.0321	0.1987	0.1893
2	0.0470	0.0328	0.1770	0.1873
3	0.0466	0.0264	0.1663	0.1655
4	0.0465	0.0268	0.1551	0.1563
5	0.0453	0.0227	0.1465	0.1316
6	0.0445	0.0232	0.1433	0.1203
7	0.0435	0.0210	0.1411	0.0955
8	0.0433	0.0211	0.1364	0.0865
9	0.0422	0.0199	0.1293	0.0700
10	0.0414	0.0202	0.1268	0.0592

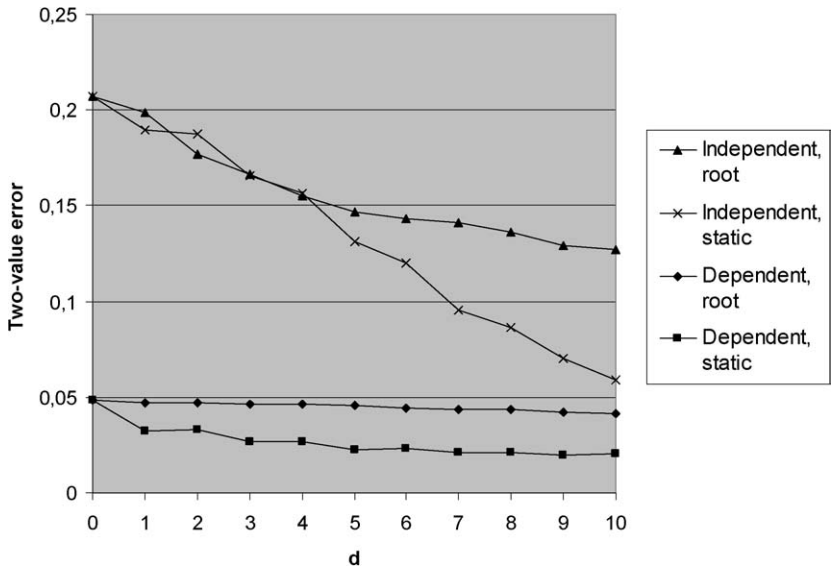

Fig. 23. Two-value error at the root and static two-value error in the dependent- and the independent-value model.

6.4. Conversion to two values

A real-value model can be reduced to a two-value model by converting the real-number values to losses and wins. To do this, a threshold must be established: the values below it are considered losses and the values above it wins. If the probability of a loss at the root is to be 0.5, the threshold should be the median backed-up real value at the root.

Table 10 and Fig. 23 show two-value error (the probability of an incorrect evaluation of a position) at the root and two-value error at the lowest ply of search, or static two-value error, with respect to the depth of search in the dependent- and the independent-value model; $b = 2$, distribution of nodes' values is normal, $m = \infty$ and $\sigma_e = 0.2$.

As can be seen in Table 10 and Fig. 23, neither the dependent- nor the independent-value model is pathological in terms of two-value error. Note that even though the error at the root is mostly larger than static error in both models, this does not mean that the models are pathological: it merely means that positions at higher plies are more difficult to evaluate as losses or wins than those at lower plies. If static error were constant, the increase of error through minimaxing would of course lead to the pathology. But as can be observed in Table 10 and Fig. 23, static two-value error is not constant, which also means that this experiment cannot be compared to the results of previous research outlined in Section 2. In order to make such a comparison possible, static position error must be adjusted so that static two-value error is constant. Table 11 and Fig. 24 show two-value error at the root with respect to the depth of search in

Table 11
Two-value error at the root in the dependent- and the independent-value model with constant static two-value error

d	Dependent-value model	Independent-value model
0	0.0202	0.0592
1	0.0306	0.0624
2	0.0277	0.0609
3	0.0354	0.0688
4	0.0349	0.0687
5	0.0391	0.0818
6	0.0390	0.0851
7	0.0436	0.0991
8	0.0402	0.1069
9	0.0438	0.1212
10	0.0414	0.1268

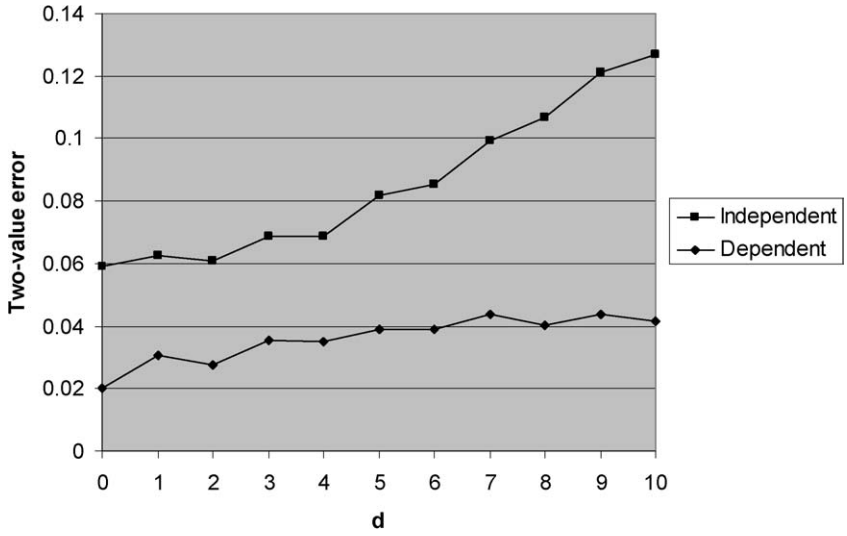

Fig. 24. Two-value error at the root of the dependent- and the independent-value model with constant static two-value error.

the dependent- and the independent-value model after this adjustment; $b = 2$, distribution of nodes' values is normal and $m = \infty$.

As can be seen in Table 11 and Fig. 24, constant static two-value error produces pathology both in the dependent- and the independent-value model; pathology is stronger in the independent-value model. This result could be interpreted as an indication that the pathology may exist even in the dependent-value model. However, the constant static two-value error assumption in this experiment is not appropriate. A constant static two-value error is indeed assumed in the two-value model in Section 2, but in a model where true values are real numbers, static two-value error should decrease with depth. The reason is that at the lowest ply of shallower searches, nodes' values are on average closer to the threshold separating losses from wins, hence two-value error is more likely to occur in such nodes. This is illustrated in Fig. 25; the darker area represents a higher probability of two-value error.

Scheucher and Kaindl [18] investigated the relation between the nodes' heuristic values and two-value error in greater detail. One of the key points of their paper was precisely that static two-value error (called simply error, since they did not consider multiple types of error) should not be constant. If static two-value error in shallower searches should indeed be larger than in deeper searches, this explains why a basic two-value model such as the one described in Section 2 is pathological: it does not model the heuristic error of a real-value game appropriately. This way of modeling error might be suitable for a two-value game, but two (or three) values are not enough for playing real games like chess successfully, as was also demonstrated by Scheucher and Kaindl.

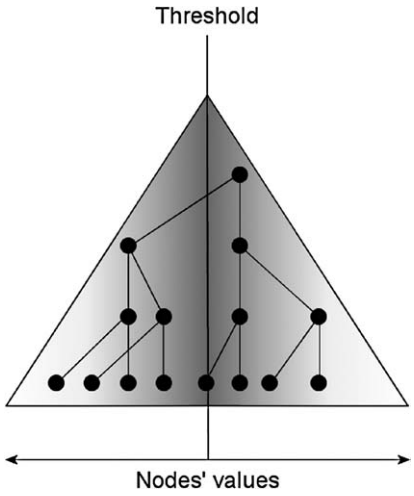

Fig. 25. Why static two-value error should not be constant.

7. Conclusion

In this paper we investigated the question whether minimax behaves pathologically in game trees where the true values of positions are real numbers. Our general conclusion is "no", which is in agreement with the practice of game-playing programs, but different from earlier theoretical analyses by several other authors who studied game trees where the true values of positions could only be losses and wins.

We constructed a minimax model where game tree nodes have real-number values and nodes' values are distributed around the value of their parent. The pathology was not present with any configuration of parameters tested.

A mathematical explanation of why minimax performs better with increased depth of search on game trees such as the ones built according to our model was provided. The reason is that minimum/maximum of pairs of random variables has a smaller variance than the variables themselves. If these random variables represent error, it is thus reduced through minimaxing. The means of the heuristic values have a slight bias with respect to the true values, but with alternating min and max plies, the bias alternates between negative and positive, preventing the heuristic values from moving too far away from the true values. Extending the theoretical analysis to trees with branching larger than two and general distribution of nodes' values around the value of their parent is left for further work.

The assumption that sibling nodes' values are normally distributed around the value of their parent was verified in the Crafty chess program. Static values of sibling nodes in the program turned out to be close to each other and their distribution around their parent's value similar to normal. This can be considered a confirmation that our model corresponds well to the situation in chess playing programs.

Considering that pathology is common in two-value models, we tried to find under what conditions, if any, it appears in real-value models. No pathology in terms of position error was found. By default, static position error was constant in our experiments. If, instead, static move error was set to be constant, our model was still not pathological, but a model with independent leaves' values did behave pathologically. This shows that similarity of nodes close to each other in a game tree can eliminate pathology.

In order to better model game-playing programs, a limited number of discrete values were assigned to positions instead of real values. Here the existence of the pathology depends on the granularity of heuristic values and on the branching factor. The chances of pathology increase with branching factor and decrease with granularity. In the dependent-value model, 19 values were sufficient to avoid the pathology in all the experiments. In these experiments, we could only reach branching factors of up to 10, but the required granularity did not rise after $b = 4$. In the dependent-value model, the required granularity increases fast with the branching factor, so a program for playing a game described by this model would require a very fine-grained evaluation function.

If real-value game trees are converted to two-value game trees, two-value error can be measured. Depending on the assumption regarding static heuristic error, dependent and independent-value models are either both pathological or both non-pathological. One assumption is that static position error (noise applied to the position values) is independent

of the depth of search. The other assumption is that static two-value error (probability of mistaking a lost position for won or vice versa) is independent of the depth of search. We will here refer to these two assumptions as assumption 1 and assumption 2 respectively. Both assumptions seem reasonable, but they generally cannot be both true in a real-value game tree. Assumption 1 does not produce pathology and is appropriate for real-value game trees. The basic two-value model of Beal [2] and others makes assumption 2, which causes pathology. This assumption is, however, not appropriate for real-value game trees. It might be suitable for two-value game trees, but playing most real-life games by two-value evaluations does not seem to be a good idea (as discussed also by Scheucher and Kaindl [18]).

Acknowledgement

This work was supported by the Slovenian Ministry of Science and Education. We would also like to thank Aleksander Sadikov, Mihael Perman and Dana S. Nau for helpful suggestions and discussion.

References

[1] I. Althöfer, Generalized minimax algorithms are no better error correctors than minimax itself, in: D.F. Beal (Ed.), Advances in Computer Chess, vol. 5, North-Holland, Amsterdam, 1989, pp. 265–282.
[2] D.F. Beal, An analysis of minimax, in: M.R.B. Clarke (Ed.), Advances in Computer Chess, vol. 2, Edinburgh University Press, Edinburgh, 1980, pp. 103–109.
[3] D.F. Beal, Benefits of minimax search, in: M.R.B. Clarke (Ed.), Advances in Computer Chess, vol. 3, Pergamon Press, 1982, pp. 17–24.
[4] I. Bratko, M. Gams, Error analysis of the minimax principle, in: M.R.B. Clarke (Ed.), Advances in Computer Chess, vol. 3, Pergamon Press, 1982, pp. 1–15.
[5] S.H. Fuller, J.G. Gaschnig, J.J. Gillogly, An analysis of the alpha-beta pruning algorithm, Technical Report, Carnegie Mellon University, 1973.
[6] R. Hyatt, Home page, http://www.cis.uab.edu/info/faculty/hyatt/hyatt.html, 2004-04-21.
[7] D.E. Knuth, R.W. Moore, An analysis of alpha-beta pruning, Artificial Intelligence 6 (4) (1975) 293–326.
[8] M. Luštrek, Minimax pathology and real-number minimax model, in: B. Zajc, A. Trost (Eds.), Proceedings of the Thirteenth International Electrotechnical and Computer Science Conference, vol. B, Portorož, Slovenia, Slovenian Section IEEE, Ljubljana, Slovenia, 2004, pp. 137–140.
[9] M. Luštrek, M. Gams, Minimaks in napaka pri ocenjavanju položajev, in: M. Bohanec, B. Filipič, M. Gams, D. Trček, B. Likar (Eds.), Proceedings A of the 6th International Multi-Conference Information Society IS 2003, 2003, pp. 89–92.
[10] D.S. Nau, Quality of decision versus depth of search on game trees, Ph.D. thesis, Duke University, 1979.
[11] D.S. Nau, An investigation of the causes of pathology in games, Artificial Intelligence 19 (3) (1982) 257–278.
[12] D.S. Nau, Decision quality as a function of search depth on game trees, J. Assoc. Comput. Mach. 30 (4) (1983) 607–708.
[13] D.S. Nau, Pathology on game trees revisited, and an alternative to minimaxing, Artificial Intelligence 21 (1, 2) (1983) 221–224.
[14] D.S. Nau, How to do worse by working harder: The nature of pathology on game trees, in: Proceedings of 1983 IEEE International Conference on Systems, Man, and Cybernetics, 1983.
[15] M.M. Newborn, The efficiency of the alpha-beta search on trees with branch-dependent terminal node scores, Artificial Intelligence 8 (2) (1977) 137–153.
[16] J. Pearl, On the nature of pathology in game searching, Artificial Intelligence 20 (4) (1983) 427–453.
[17] A. Sadikov, I. Bratko, I. Kononenko, Search vs knowledge: Empirical study of minimax on KRK endgame, in: H.J. van den Herik, H. Iida, E. Heinz (Eds.), Advances in Computer Games: Many Games, Many Challenges, Kluwer Academic, Dordrecht, 2003, pp. 33–44.
[18] A. Scheucher, H. Kaindl, Benefits of using multivalued functions for minimaxing, Artificial Intelligence 99 (2) (1998) 187–208.
[19] G. Schrüfer, Presence and absence of pathology on game trees, in: D.F. Beal (Ed.), Advances in Computer Chess, vol. 4, Pergamon Press, 1986, pp. 101–112.
[20] B. Sjörgen, Swedish Chess Computer Association website, http://w1.859.telia.com/~u85924109/ssdf/, 2004-04-21.

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Artificial Intelligence 170 (2006) 643–652
Artificial Intelligence
www.elsevier.com/locate/artint

Existential assertions and quantum levels on the tree of the situation calculus

Francesco Savelli

Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, I-00198 Roma, Italy

Received 14 February 2005; received in revised form 7 December 2005; accepted 5 January 2006

Available online 8 February 2006

Abstract

In a seminal paper, Reiter introduced a variant of the situation calculus along with a set of its properties. To the best of our knowledge, one of these properties has remained unproved and ignored despite its relevance to the planning problem and the expressivity of the theories of actions. We state this property in a more general form and provide its proof. Intuitively, whenever a theory of actions entails that there exists a situation satisfying a first order formula (e.g., a goal), at least one such situation must be found within a predetermined distance from the initial situation. This distance is *finite* and the *same* in all the models of the theory, since it depends only on the theory and the formula at hand.

Keywords: Knowledge representation; Reasoning about actions and change; Situation calculus

1. Introduction

In reasoning about actions and change with partial knowledge, computing a plan of actions that achieves a specified goal can be impossible even if the existence of the plan is formally guaranteed. Indeed, if a logical theory of actions does not include a complete axiomatic representation of the system's state, no one plan (or finite disjunction of plans) might exist that simultaneously meets the goal in all the models of the theory, though each model might still contain its own valid plan. When this is the case, no finite plan guaranteed to succeed can be built and executed in the real world, even though its existence is actually entailed by the theory.

This problem has a theoretical solution in the variant of the situation calculus [1] introduced by Reiter [2], which has represented one of the most influential approaches to laying formal foundations for planning, diagnosis, and agent specification in artificial intelligence, besides having contributed new insights into database theory. For nearly fifteen years, a diverse body of research has stemmed from Reiter's seminal paper [2], and the original ideas and notions contained therein are currently at the core of much work in cognitive robotics. All but one of the properties of the formalism were formally proved in a later paper [3] after cleaner and more systematic axiomatization, and have been extensively employed in later research.

E-mail address: Francesco.Savelli@dis.uniroma1.it (F. Savelli).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2006.01.002

To the best of our knowledge, the theorem in [2] that provides the solution to the problem introduced above has never been mentioned again, and its formal justification has not appeared in literature. The theorem formally guarantees that if a theory of actions entails the existence of a plan that achieves a given goal, a finite disjunctive instance can be built.

This result is still theoretically very relevant as the research community begins to investigate more complex systems, and consequently the problem of plan computability is being recasted in more sophisticated terms. One noteworthy example is the problem of *"knowing how"* to execute a program augmented with non-deterministic choices and sensing actions [4,5] in cognitive robotics.

In this paper we provide the missing proof of Reiter's theorem. We state the result in more general terms and show its implications for the expressivity of the formalism. Intuitively, whenever an action theory entails that there exists a situation satisfying a first order formula (e.g., a goal), at least one such situation must be found inside a special region of the tree of situations that characterizes the models of the theory. This region takes the form of a *finite* subset of all the levels of the tree, and it is the *same* in all the models of the theory, since it depends only on the theory and the formula at hand. We dub the situation-tree levels belonging to this subset *"quantum levels"*.

The rest of this paper is organized as follows. In Section 2 we recall the basic formal notions needed to make this paper self-contained, and introduce some preliminary definitions and results that are straightforward consequences of previous work. In Section 3 we state and prove some properties of the situation calculus, including the theorem mentioned above (Theorem 2 of [2]). In Section 4 we discuss the implications of these results for the expressivity of the situation calculus and for current research in reasoning about actions and change.

2. The situation calculus

We assume the reader is familiar with the basic concepts of classical logic [6]. We recall some notations.

In a formula, by τ_1/τ_2 we denote that the occurrences of τ_1 are replaced by τ_2. By $(\forall)$ or $(\exists)$ at the beginning of a formula we intend that all the free variables occurring in the formula are universally or existentially closed. Given a function or constant symbol ω of a language $\mathcal{L}$, $\omega^{\mathbb{M}}$ denotes the interpretation of ω in the structure $\mathbb{M}$ of $\mathcal{L}$. We extend the classical notation of entailment to set of sentences: $\mathbb{M} \models \{\phi_i\}_{i\in\mathbb{N}}$ means that the structure $\mathbb{M}$ entails every sentence $\phi_1, \phi_2, \ldots$ of the set simultaneously. The meaning of $T \models \{\phi_i\}_{i\in\mathbb{N}}$, where T is a theory, is likewise defined.

We use the notations, definitions, and results given in [3], which provides systematic and formally complete foundations for the situation calculus as formulated in [2]. These notions are recalled in Sections 2.1, 2.2, and 2.3. Sections 2.4, 2.5, and 2.6 introduce notions that simply follow from previous results.

2.1. The language $\mathcal{L}_{sitcalc}$

The language $\mathcal{L}_{sitcalc}$ is a three-sorted second order language with equality. The three sorts are *action* for actions, *situation* for situations, and a catch-all sort *object* for everything else depending on the domain of application.

We adopt the following notations, with subscripts and superscripts: α and a for terms and variables of sort *action*, σ and s for terms and variables of sort *situation*; t and x for terms and variables of any sort.

S_0 is the constant denoting the initial situation, and

$$do : action \times situation \to situation$$

is a special function that enables the inductive construction of successive situations. The term $do(\alpha, \sigma)$ interprets the situation resulting from performing the action α in the situation σ. Thus the term

$$do(\alpha_n, do(\ldots, do(\alpha_2, do(\alpha_1, S_0))))$$

—abbreviated as $do([\alpha_1, \ldots, \alpha_n], S_0)$—interprets the situation obtained after the sequence of actions $\alpha_1, \ldots, \alpha_n$ has taken place. (If $n = 0$, $do([\alpha_1, \ldots, \alpha_n], \sigma)$ is σ.)

Hence, situations represent sequences of actions, which should be viewed as histories of the dynamical domain, rather than as its states. This is a fundamental distinguishing feature of Reiter's variant of the situation calculus.

The actions are usually completely represented by a finite set of *action function symbols* with signature $(action \cup object)^n \to action$. As an example, consider the action function symbol $moveOn(x, y)$ denoting the action of moving

the object x on the object y. The term $do(moveOn(book, table), S_0)$ will interpret the situation in which this action has been applied to the two objects interpreted by the constant symbols *book* and *table*.

Two special predicates are $\sqsubset : situation \times situation$, which defines an ordering relation on situations, and $Poss : action \times situation$, which defines which actions can be performed in a given situation.

The dynamical properties of the domain are represented by function and predicate symbols with signatures $(action \cup object)^n \times situation \to (action \cup object)$ and $(action \cup object)^n \times situation$. These functions and predicates are respectively called *functional* and *relational fluents*. Following the example above, the relational fluents $On(x, y, s)$ and $Holding(z, s)$ may mean that in the situation s the object x is on the object y, and that the object z is being held by the agent. Non-fluent functions or predicates can be used to represent properties and facts independent of any specific situation, like $Flat(table)$ or $Larger(table, book)$.

The use of terms of sort *situation* in $\mathcal{L}_{sitcalc}$ is limited. The only constant and function symbols of this sort are S_0 and do. The only predicates allowed to take an argument of sort *situation* are *Poss*, $\sqsubset$, the equality, and the relational fluents. The only functions allowed to take an argument of sort *situation* are do and the functional fluents.

In a structure $\mathbb{M}$ of $\mathcal{L}_{sitcalc}$ the domain is partitioned into the subsets *Obj*, *Act*, and *Sit*, reflecting the three sorts of the language. Terms of sorts *object*, *action*, and *situation* range respectively over these different domains. We use ν for an assignment to free variables of any sort.

2.2. Basic action theories

A formula of $\mathcal{L}_{sitcalc}$ is *uniform* in σ iff it is first order, it does not mention the predicates *Poss* or $\sqsubset$, it does not mention equality on situations, it does not quantify over variables of sort *situation*, and σ is the only term of sort *situation* that is mentioned in the fluents occurring in the formula. The set of these formulas can be syntactically defined by induction [3]. Intuitively, the key property of a formula uniform in σ is that it "concerns only σ".

A *basic action theory* $\mathcal{D}$ is composed of the foundational axioms Σ, the set of action precondition axioms $\mathcal{D}_{ap}$, the set of successor state axioms $\mathcal{D}_{ss}$, the set of unique names axioms for the action function symbols $\mathcal{D}_{una}$, and the initial database $\mathcal{D}_{S_0}$. The structure of these sets and of their axioms are as follows.[1]

- Σ contains four *foundational, domain-independent axioms*.

  $$(\forall)\ do(a_1, s_1) = do(a_2, s_2) \supset a_1 = a_2 \wedge s_1 = s_2 \tag{1}$$

  is a unique name axiom for situations.

  $$(\forall P)\ \{P(S_0) \wedge (\forall a, s)[P(s) \supset P(do(a, s))]\} \supset (\forall s) P(s) \tag{2}$$

  is a second order induction axiom on situations, whose purpose is very similar to that of the analogous axiom in Peano's arithmetic [6]. Axiom (2) has the important effect of limiting the domain *Sit* to the smallest set that (i) contains the interpretation of S_0, and (ii) is closed under the application of the function do to the elements of *Act* and recursively of *Sit* itself. This inductive definition constructs an infinite tree of situations that is rooted in S_0 and contains all the elements of *Sit* as its nodes. Every node branches on all the elements of *Act* [7].
  The last two axioms define the ordering relation on situations:

  $$(\forall s)\ \neg s \sqsubset S_0 \tag{3}$$

  $$(\forall s, s', a)\ s \sqsubset do(a, s') \equiv s \sqsubset s' \vee s = s' \tag{4}$$

  Intuitively, $s \sqsubset s'$ means that s' can be obtained from s by performing an appropriate sequence of actions.
- $\mathcal{D}_{ap}$ contains one *action precondition axiom* in the form

  $$(\forall x_1, \ldots, x_n, s)\ Poss(A(x_1, \ldots, x_n), s) \equiv \Pi_A(x_1, \ldots, x_n, s) \tag{5}$$

  for each action function symbol A of $\mathcal{L}_{sitcalc}$, where $\Pi_A(x_1, \ldots, x_n, s)$ is a formula uniform in s that does not mention any free variable other than $x_1, \ldots, x_n, s$.

[1] For ease of exposition, in this section we will not treat the functional fluents. The complete introduction requires a straightforward technical extension. The interested reader is referred to [3,7]. The theorems provided in this paper and their proofs are formally complete with respect to the full framework including functional fluents.

An action precondition axiom defines the preconditions to the executability of an action in a given situation, in terms of properties holding in that situation alone. Following the simple example above, we may define:

$$(\forall x, s)\ Poss(pickUp(x), s) \equiv \neg(\exists y) Holding(y, s)$$
$$(\forall x, s)\ Poss(putDown(x), s) \equiv Holding(x, s)$$
$$(\forall x, y, s)\ Poss(moveOn(x, y), s) \equiv Holding(x, s) \wedge Flat(y) \wedge Larger(y, x)$$

- $\mathcal{D}_{ss}$ contains one *successor state axiom* in the form

$$(\forall x_1, \ldots, x_n, a, s)\ F(x_1, \ldots, x_n, do(a, s)) \equiv \Phi_F(x_1, \ldots, x_n, a, s) \tag{6}$$

for each relational fluent F of $\mathcal{L}_{sitcalc}$, where $\Phi_F(x_1, \ldots, x_n, a, s)$ is a formula uniform in s that does not mention any free variable other than $x_1, \ldots, x_n, a, s$.
A successor state axiom defines how a fluent's truth value changes from situation to situation, based only on the action taken and on the properties holding in the current situation. For example:

$$(\forall x, y, a, s)\ On(x, y, do(a, s)) \equiv a = moveOn(x, y) \vee [On(x, y, s) \wedge a \neq pickUp(x)]$$
$$(\forall x, a, s)\ Holding(x, do(a, s)) \equiv a = pickUp(x) \vee [Holding(x, s) \wedge a \neq putDown(x)]$$

- $\mathcal{D}_{una}$ contains *unique name axioms for the action function symbols*. For every two distinct action function symbols A and B of $\mathcal{L}_{sitcalc}$:

$$(\forall)\ A(x_1, \ldots, x_n) \neq B(y_1, \ldots, y_m)$$

Besides, for each action function symbol A:

$$(\forall)\ A(x_1, \ldots, x_n) = A(y_1, \ldots, y_n) \supset x_1 = y_1 \wedge \cdots \wedge x_n = y_n$$

- $\mathcal{D}_{S_0}$, often called the *initial database*, is a set of first order sentences that are uniform in S_0. Their function is to describe the world as it is before any action has been executed. Some of these sentences may mention no situation, to the purpose of representing situation-independent properties. Unique name axioms for individuals are a typical case of such sentences.
An initial database for our simple example might be:

$$\neg(\exists x)\ Holding(x, S_0)$$
$$table \neq book$$
$$Flat(table)$$
$$Larger(table, book)$$

Note that a basic action theory thus defined is first order except for the induction axiom (2).

2.3. Relative satisfiability and regression

Relative satisfiability is a key property of basic action theories that will play an essential role in this paper.

Theorem 1 *(Relative satisfiability). A basic action theory $\mathcal{D}$ is satisfiable iff $\mathcal{D}_{S_0} \cup \mathcal{D}_{una}$ is.*

Regression is a form of automated reasoning widely employed in computational procedures for planning and diagnosis. There is a precise and natural way to characterize regression formally in the situation calculus. We first need to define the class of the regressable formulas of $\mathcal{L}_{sitcalc}$.

Definition 2 *(The regressable formulas).* A formula W of $\mathcal{L}_{sitcalc}$ is *regressable* iff

(1) W is first order.

(2) Every term of sort *situation* mentioned by W has the form $do([\alpha_1, \ldots, \alpha_n], S_0)$ for some[2] $n \geqslant 0$ and terms $\alpha_1, \ldots, \alpha_n$ of sort *action*.
(3) For every term of the form $Poss(\alpha, \sigma)$ mentioned by W, α has the form $A(t_1, \ldots, t_n)$ for some action function symbol A of $\mathcal{L}_{sitcalc}$, $n \geqslant 0$, and terms $t_1, \ldots, t_n$ of appropriate sorts.

Observe that any formula uniform in σ is trivially regressable if σ is not a variable of sort *situation*.

Given a basic action theory $\mathcal{D}$ and a regressable formula W, the *regression operator* $\mathcal{R}$ applied to W generates a formula $\mathcal{R}[W]$ uniform in S_0 that is logically equivalent to W in the models of $\mathcal{D}$. This operator is defined by induction both on the syntactic structure of W and on the number of actions in the terms of sort *situation* occurring in W. This definition as well as a complete and detailed account of all the syntactic transformations involved by $\mathcal{R}$ can be found elsewhere [3,7]. We emphasize here that such transformations are chiefly based on employing the logical definitions (equivalences) embodied by the successor state axioms as rewriting rules. In this way, the number of actions in the terms of sort *situation* can be reduced. Since the rewriting rules are axioms of $\mathcal{D}$, logical equivalence is preserved. After a finite number of iterations, this process ends with the formula $\mathcal{R}[W]$ uniform in S_0.

A formal account of the properties of $\mathcal{R}$ is given by two theorems.

Theorem 3. *Let W be a regressable formula of $\mathcal{L}_{sitcalc}$ and $\mathcal{D}$ a basic action theory. Then $\mathcal{R}[W]$ is uniform in S_0, and*

$$\mathcal{D} \models (\forall)\ (W \equiv \mathcal{R}[W])$$

By combining Theorems 1 and 3 the following important result is obtained [3] as well:

Theorem 4 *(Soundness and completeness of regression). Let W be a regressable formula of $\mathcal{L}_{sitcalc}$ and $\mathcal{D}$ a basic action theory. Then*

$$\mathcal{D} \models W \quad \textit{iff} \quad \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[W]$$

2.4. Executable situations

A logical definition of the executable situations is provided [7] by the recursive macro:

$$ex(s) \stackrel{def}{\equiv} s = S_0 \vee (\exists a, s')\ s = do(a, s') \wedge Poss(a, s') \wedge ex(s')$$

For a term $do([\alpha_1, \ldots, \alpha_n], S_0)$ of sort *situation*, in the models of Σ it follows that

$$ex(do([\alpha_1, \ldots, \alpha_n], S_0)) \equiv Poss(\alpha_1, S_0) \wedge Poss(\alpha_2, do(\alpha_1, S_0)) \wedge \cdots \wedge Poss(\alpha_n, do([\alpha_1, \ldots, \alpha_{n-1}], S_0)) \tag{7}$$

2.5. Closure on actions

In Theorem 10 we shall consider a basic action theory that includes the following action closure property as an axiom:

$$(\forall a) \quad \bigvee_{i=1\ldots q} (\exists x_1, \ldots, x_{n_i})\ a = A_i(x_1, \ldots, x_{n_i}) \tag{8}$$

where $A_1, \ldots, A_q$ are all the action function symbols for which action precondition axioms are provided in $\mathcal{D}_{ap}$.

Note that for such a basic action theory, given a formula $\theta(a)$ in which the variable a of sort action occurs free, the following holds

$$\mathcal{D} \models [(\forall a)\ \theta(a)] \equiv \bigwedge_{i=1\ldots q} (\forall x_1, \ldots, x_{n_i})\ \theta(a/A_i(x_1, \ldots, x_{n_i})) \tag{9}$$

[2] Different situation terms are typically mentioned by W, that is, n and $\alpha_1, \ldots, \alpha_n$ vary across the terms $do([\alpha_1, \ldots, \alpha_n], S_0)$ occurring in W. If this is not the case, W can be shown equivalent to a formula uniform in $do([\alpha_1, \ldots, \alpha_n], S_0)$.

2.6. Compactness of basic action theories

Compactness is a property of first order logic theories that in general does not extend to higher order languages. Basic action theories are a fortunate case in which compactness does hold, although they include the second order induction axiom (2) in Σ. This can be stated as a corollary of the relative satisfiability (Theorem 1):

Corollary 5 *(Compactness of basic action theories). Let $\mathcal{D}$ be an infinite basic action theory. $\mathcal{D}$ is unsatisfiable iff there exists a first order finite unsatisfiable subset of $\mathcal{D}$.*

Proof. The *if* part is straightforward. Let us address the *only if*.

$\mathcal{D}$ is unsatisfiable. Let $\mathcal{D}\_$ be the subtheory obtained when the second order induction axiom is removed from $\mathcal{D}$. If $\mathcal{D}\_$ were satisfiable, its subtheory $\mathcal{D}_{S_0} \cup \mathcal{D}_{una}$ would be satisfiable too, and so would $\mathcal{D}$ be by relative satisfiability, which contradicts the hypothesis. Then $\mathcal{D}\_$ must be unsatisfiable and, since it is a first order theory, it is also compact. Therefore there must exist an unsatisfiable finite subset of $\mathcal{D}\_$. □

3. Quantum levels and Reiter's theorem

As recalled in Section 2.2, the domain *Sit* in the models of Σ has the form of a tree rooted in S_0 whose nodes and arcs are respectively the elements of the domains *Sit* and *Act* [7]. There are countably infinitely many levels for this tree. This does not imply that *Sit* is countable, as the branching factor at every node is equal to the cardinality of *Act*; since no constraint for this is commonly given, in the general case *Sit* will take any infinite cardinality across all the models, due to the Lowenheim–Skolem–Tarski theorem [6]. However, if we split a quantification over situations into a double quantification, the first over the levels and the second over the situations belonging to the current level, the former can be syntactically expanded thanks to the countability of the tree's levels. This is formally stated and justified in Lemma 6 below.

First, note that for a formula $\varphi(s)$, in which s is a variable of sort *situation* occurring free, the universal and existential quantifications over the situations belonging to the level l of the situation tree have the form

$$(\forall a_1, \ldots, a_l)\ \varphi(s/do([a_1, \ldots, a_l], S_0))$$

and

$$(\exists a_1, \ldots, a_l)\ \varphi(s/do([a_1, \ldots, a_l], S_0))$$

where $a_1, \ldots, a_l$ are fresh variable symbols of sort *action*.

Lemma 6. *Let $\varphi(s)$ be a formula of $\mathcal{L}_{sitcalc}$, where s is a variable of sort situation occurring free. Let $\mathbb{M}$ be a model of Σ and v an assignment to the free variables occurring in $\varphi(s)$ except s, then*

$$\mathbb{M}, v \models (\forall s)\ \varphi(s) \quad \textit{iff} \quad \mathbb{M}, v \models \{(\forall a_1, \ldots, a_l)\ \varphi(s/do([a_1, \ldots, a_l], S_0))\}_{l \in \mathbb{N}}$$

where $a_1, \ldots, a_l$ are fresh variable symbols of sort action.

Proof. The *only if* part is straightforward. Let us address the *if*.

$$\mathbb{M}, v \models \Sigma \cup \{(\forall a_1, \ldots, a_l)\ \varphi(s/do([a_1, \ldots, a_l], S_0))\}_{l \in \mathbb{N}}. \tag{10}$$

Assume

$$\mathbb{M}, v \not\models (\forall s)\ \varphi(s)$$

that is

$$\mathbb{M}, v \models (\exists s)\ \neg\varphi(s) \tag{11}$$

Now pick a witness in $\mathbb{M}$ for (11), i.e., an element of the domain *Sit* of $\mathbb{M}$, say $\dot{s}$, such that $\varphi^{\mathbb{M},v}(\dot{s})$ is false. Since $\mathbb{M} \models \Sigma$ we know that every situation corresponds to a unique path on the situation tree, which is the sequence of

elements of *Act* connecting the situation to the root of the tree S_0. Formally, it is $\dot{s} = do^{\mathbb{M}}([\dot{a}_1, \ldots, \dot{a}_p], S_0^{\mathbb{M}})$ for some $p \in \mathbb{N}$ and some $\dot{a}_1, \ldots, \dot{a}_p$ elements of the domain *Act* of $\mathbb{M}$. Therefore

$$\mathbb{M}, \nu \not\models (\forall a_1, \ldots, a_p)\ \varphi(s/do([a_1, \ldots, a_p], S_0))$$

which contradicts the initial assumption (10). □

We can now provide the main result of this paper.

Theorem 7. *Let $\mathcal{D}$ be a satisfiable basic action theory and $\varphi(s)$ a first order formula of $\mathcal{L}_{sitcalc}$ such that, for any $n \in \mathbb{N}$ and $\alpha_1, \ldots, \alpha_n$ terms of sort action, the formula $\varphi(s/do([\alpha_1, \ldots, \alpha_n], S_0))$ is regressable.*[3] *Then*

$$\mathcal{D} \models (\forall)\ (\exists s)\ \varphi(s)$$

iff there exists a finite subset Λ of $\mathbb{N}$ such that

$$\mathcal{D} \models \bigvee_{l \in \Lambda} (\forall)\ (\exists a_1, \ldots, a_l)\ \varphi(s/do([a_1, \ldots, a_l], S_0))$$

where $a_1, \ldots, a_l$ are fresh variable symbols of sort action.

Proof. The *if* part is straightforward. Let us address the *only if*.

$\mathcal{D} \models (\forall)\ (\exists s)\ \varphi(s)$. This can also be stated as

$$\mathcal{D} \cup \{(\exists)\ (\forall s)\ \neg\varphi(s)\} \text{ is unsatisfiable.}$$

By Lemma 6 the theory

$$\mathcal{D} \cup \{(\exists)\ (\forall a_1, \ldots, a_l)\ \neg\varphi(s/do([a_1, \ldots, a_l], S_0))\}_{l \in \mathbb{N}}$$

must be unsatisfiable too. Every sentence

$$(\exists)\ (\forall a_1, \ldots, a_k)\ \neg\varphi(s/do([a_1, \ldots, a_k], S_0))$$

can be regressed (Theorem 3) into a sentence uniform in S_0, say ψ_k, such that

$$\mathcal{D} \models [(\exists)\ (\forall a_1, \ldots, a_k)\ \neg\varphi(s/do([a_1, \ldots, a_k], S_0))] \equiv \psi_k \tag{12}$$

Therefore, the set

$$\mathcal{D}' = \mathcal{D} \cup \{\psi_l\}_{l \in \mathbb{N}}$$

is an unsatisfiable theory. In particular, by letting

$$\mathcal{D}'_{S_0} = \mathcal{D}_{S_0} \cup \{\psi_l\}_{l \in \mathbb{N}}$$

it is easy to observe that $\mathcal{D}'$ formally amounts to a new infinite basic action theory

$$\Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}'_{S_0}$$

Then, by compactness (Corollary 5), there must exist an unsatisfiable finite subset $\mathcal{D}'_f$ of $\mathcal{D}'$. Since $\mathcal{D}$ is satisfiable by hypothesis, $\mathcal{D}'_f$ must contain some (finitely many) elements of $\{\psi_l\}_{l \in \mathbb{N}}$: let Λ be the finite subset of $\mathbb{N}$ containing their l indexes. The finite superset of $\mathcal{D}'_f$

$$\mathcal{D} \cup \{\psi_l\}_{l \in \Lambda}$$

must be unsatisfiable too.

Now, reasoning backward, by (12)

$$\mathcal{D} \cup \{(\exists)\ (\forall a_1, \ldots, a_l)\ \neg\varphi(s/do([a_1, \ldots, a_l], S_0))\}_{l \in \Lambda}$$

[3] Hence, s is the only variable of sort situation in $\varphi(s)$ and occurs free, and no variable of sort situation is mentioned by any term α_i.

is unsatisfiable, that is

$$\mathcal{D} \models \bigvee_{l \in \Lambda} (\forall)\ (\exists a_1, \ldots, a_l)\ \varphi(s/do([a_1, \ldots, a_l], S_0)). \qquad \square$$

This result points out how the expressive power of the situation calculus is affected by the compactness of the basic action theories, which in turn is due to the compactness of the first order theory $\mathcal{D}_{S_0}$ (see Corollary 5). For example, a particular property cannot come true in only one situation whose corresponding sequence of actions has a different length in every model of the theory. More generally, if a basic action theory entails the existence of a situation with a given property, this situation cannot range over the levels of the situation tree in full freedom. Indeed the basic action theory coupled with the property determines what we call the "quantum levels" of the situation tree: given $\mathcal{D}$ and $\varphi(s)$ such that $\mathcal{D} \models (\exists s)\ \varphi(s)$, there must exist a finite subset of levels of the situation tree, *depending only on* $\mathcal{D}$ *and* φ, on which at least one witness for $(\exists s)\ \varphi(s)$ is found. Note that this set is the same in *every model of* $\mathcal{D}$.

A proof of Reiter's Theorem 2 in [2]—rephrased below according to the notations and axiomatization in [3] followed here—can be easily obtained as a variant of the proof of Theorem 7 above. We first introduce some convenient definitions:

Definition 8. Let $\mathcal{D}$ be a basic action theory that additionally includes the action closure axiom (8). Let $k \in \mathbb{N}$, and $G(do([a_1, \ldots, a_k], S_0))$ be a first order formula of $\mathcal{L}_{sitcalc}$ uniform in $do([a_1, \ldots, a_k], S_0)$. Ψ_k is the formula obtained after all the occurrences of the variables $a_1, \ldots, a_k$ and their quantifications in

$$(\forall a_1, \ldots, a_k)\ \neg\left[G(do([a_1, \ldots, a_k], S_0)) \wedge \bigwedge_{i=1\ldots k} Poss(a_i, do([a_1, \ldots, a_{i-1}], S_0))\right] \tag{13}$$

have been replaced by applying the equivalence (9) k times.

After this transformation the predicate $Poss(\ldots)$ occurs consistently with Definition 2 of regressable formulas. The remaining subformula expanded from $G(\ldots)$ inherits regressability from $G(\ldots)$, which is regressable because it is uniform in $do([a_1, \ldots, a_k], S_0)$. Ψ_k is hence regressable, thus let

$$\Gamma_k = \neg\mathcal{R}[\Psi_k]$$

The purpose of the previous definition is to provide a regressable formula equivalent to (13). Indeed $Poss(a_i, do([a_1, \ldots, a_{i-1}], S_0))$ is not in a regressable form (Definition 2). We can then rely on a chain of equivalences:

Proposition 9. *For Definition* 8 *we have*:

$$\mathcal{D} \models (\forall a_1, \ldots, a_k)\ \neg[G(do([a_1, \ldots, a_k], S_0)) \wedge ex(do([a_1, \ldots, a_k], S_0))] \equiv$$

(*by* (7))

$$(\forall a_1, \ldots, a_k)\ \neg\left[G(do([a_1, \ldots, a_k], S_0)) \wedge \bigwedge_{i=1\ldots k} Poss(a_i, do([a_1, \ldots, a_{i-1}], S_0))\right] \equiv$$

(*by Definition* 8)

$$\Psi_k \equiv$$

(*by Definition* 8 *and Theorem* 3)

$$\neg\Gamma_k$$

Theorem 10. *Let* $\mathcal{D}$ *be a satisfiable basic action theory that additionally includes the action closure axiom* (8)*. Let* $G(s)$ *be a first order formula of* $\mathcal{L}_{sitcalc}$ *uniform in the variable of sort situations, which is also the only variable occurring free. Let* Γ_i*,* $i \in \mathbb{N}$*, be the sentences produced from* $G(s/do([a_1, \ldots, a_i], S_0))$ *as in Definition* 8*. Then*

$$\mathcal{D} \models (\exists s)\ [G(s) \wedge ex(s)]$$

iff for some $n \in \mathbb{N}$

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \bigvee_{i=1\ldots n} \Gamma_i$$

Proof. We address the *if* part, which is rather straightforward by Definition 8 and Proposition 9.

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \bigvee_{i=1\ldots n} \Gamma_i$$

By Proposition 9,

$$\mathcal{D} \models \bigvee_{i=1\ldots n} \neg\Psi_i$$

and, again by Proposition 9,

$$\mathcal{D} \models \bigvee_{i=1\ldots n} (\exists a_1, \ldots, a_i)\ [G(do([a_1, \ldots, a_i], S_0)) \wedge ex(do([a_1, \ldots, a_i], S_0))]$$

from which it follows

$$\mathcal{D} \models (\exists s)\ [G(s) \wedge ex(s)]$$

For the *only if* part the proof proceeds as for Theorem 7, provided some additional care for "$ex(s)$" as follows.

Let $\varphi(s) = G(s) \wedge ex(s)$; after the first step of the proof of Theorem 7 we had that the theory

$$\mathcal{D} \cup \{(\forall a_1, \ldots, a_l)\ \neg\varphi(s/do([a_1, \ldots, a_l], S_0))\}_{l\in\mathbb{N}}$$

is unsatisfiable.

For the next step the formula in curly brackets should be in regressable form; to this purpose we can rely on Proposition 9

$$\mathcal{D} \cup \{\Psi_l\}_{l\in\mathbb{N}} \text{ is unsatisfiable}$$

Proceeding as for Theorem 7, it can be shown that there exists a finite subset Λ of $\mathbb{N}$ such that

$$\mathcal{D} \models \bigvee_{l\in\Lambda} \neg\Psi_l$$

By Theorem 4 and Proposition 9, we have

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \bigvee_{l\in\Lambda} \Gamma_l$$

By choosing $n = \max \Lambda$, we can conclude

$$\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \bigvee_{i=1\ldots n} \Gamma_i \qquad \square$$

4. Conclusion

This paper contributes the proof of a key property of Reiter's variant of the situation calculus. The property was first stated as a theorem in a seminal paper [2], but its proof was not provided. We have rephrased and proved the theorem in a more general form. To this end, we have pointed out that this result is a consequence of the compactness of the situation calculus' basic action theories, which significantly affects the expressivity of the formalism.

The theorem constrains how the situations satisfying an existential assertion can take place over the levels of the situation tree. If one or more situations exist in which a given property (e.g., a goal) is satisfied, at least one of these situations must be found on certain levels of the tree, within a finite distance from the initial situation. The set of such levels and the distance is the same in every model. This is a key formal guarantee for generating a plan as a finite syntactic term of the language when distinct models contain different plan solutions, as may be the case when knowledge is incomplete.

This important property of the situation calculus may not hold if a basic action theory does not comply with some of the constraints assumed in [2], and later required by definition in [3].

For instance, if the initial database ($\mathcal{D}_{S_0}$) is not first order (Section 2.2), Theorems 7 and 10 may not hold, since their proofs rest on the compactness of the basic action theories, which in turn is guaranteed by the fact that the initial

database is first order (Corollary 5). This case arises in the formalization of the chop problem [4,5], which is a simple action domain showing that an agent might not know how to execute a program with non-deterministic choices and sensing actions, even though its executability is formally guaranteed by the underlying basic action theory known to the agent. This basic action theory is purposely designed so that the first situation in which the goal is reached is at an arbitrary distance from the initial situation in every model. It follows that there can be no such quantum levels as predicted by Theorem 7, nor can there be such a finite number n as predicted by Theorem 10. This is only possible because the hypotheses of the two theorems do not hold: the initial database is second order. Indeed, since quantifying over the natural numbers is essential to the correctness of the chop-problem formalization, the exact standard model of $\mathbb{N}$ must be selected. The second order induction axiom of Peano's axiomatization of arithmetic is therefore required in the initial database, in order to rule out the non-standard models containing Z-chains [6].

The chop problem generalizes the problem of building a plan that is known to exist according to a given basic action theory—which was originally addressed by Reiter's theorem. The insight gained here is that this generalization, too, falls within the scope of the theorem; the chop problem, or one with a similar structure, cannot arise in basic action theories with a first order initial database.

Acknowledgements

I am grateful to Fiora Pirri for her stimulating discussion and for her comments on an earlier draft. I would like to thank Giuseppe De Giacomo for acquainting me with the chop problem. Two anonymous referees have provided valuable suggestions for improving this paper.

References

[1] J. McCarthy, Situations, actions and causal laws, in: M. Minsky (Ed.), Semantic Information Processing, MIT Press, Cambridge, MA, 1968, pp. 410–417.
[2] R. Reiter, The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression, in: V. Lifschitz (Ed.), Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy, Academic Press, San Diego, CA, 1991, pp. 359–380.
[3] F. Pirri, R. Reiter, Some contributions to the metatheory of the situation calculus, J. ACM 46 (3) (1999) 261–325.
[4] Y. Lespérance, H. Levesque, F. Lin, R. Scherl, Ability and knowing how in the situation calculus, Studia Logica 66 (1) (2000) 165–186.
[5] S. Sardina, G. De Giacomo, Y. Lespérance, H. Levesque, On ability to autonomously execute agent programs with sensing, in: Proceedings of the 4th International Workshop on Cognitive Robotics (CoRobo-04), Valencia, Spain, 2004, pp. 88–93.
[6] H.B. Enderton, A Mathematical Introduction to Logic, Academic Press, San Diego, CA, 1972.
[7] R. Reiter, Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems, MIT Press, Cambridge, MA, 2001.